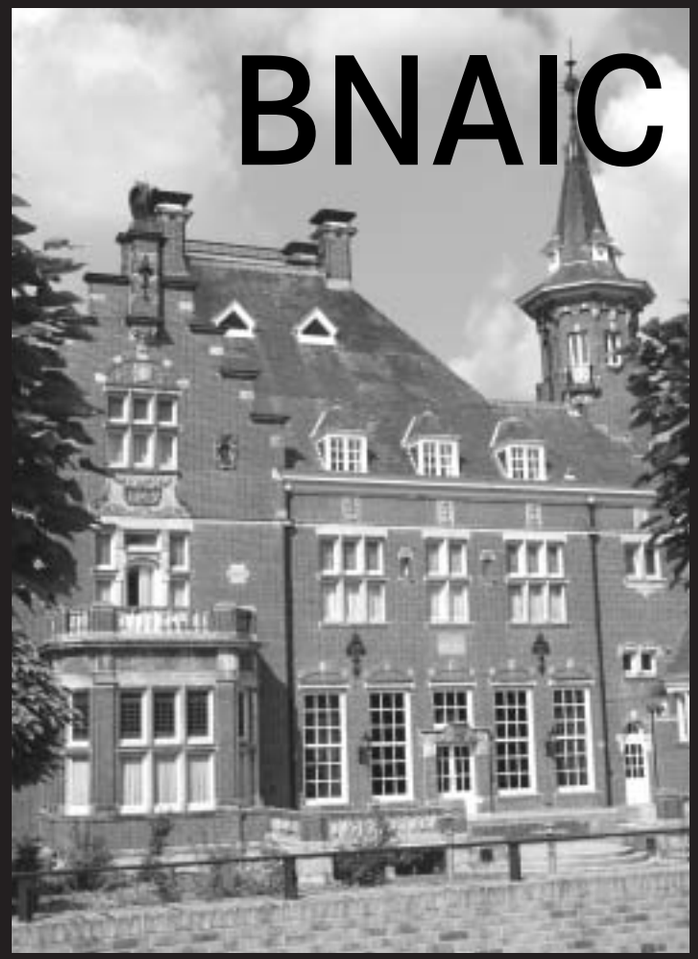
BNAIC 2003

editors:
Tom Heskes I Peter Lucas
Louis Vuurpijl I Wim Wiegerinck

BNAIC 2003



editors:
Tom Heskes
Peter Lucas
Louis Vuurpijl
Wim Wiegerinck

15th Belgium-Netherlands Conference on Artificial Intelligence

October 23-24, 2003, Nijmegen

BNAIC'03 Sponsors

BNVKI

Elsevier

KNAW

NICI

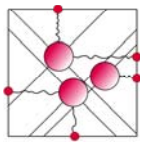
NWO

SIKS

SKBS

SPSS

SNN



University Nijmegen

BNAIC'03

Proceedings of the 15th Belgium-Netherlands
Conference on Artificial Intelligence

University of Nijmegen
23-24 October, 2003

Tom Heskes
Peter Lucas
Louis Vuurpijl
Wim Wiegerinck
(Eds.)

Preface

This book contains the proceedings of BNAIC'03, the fifteenth Belgian-Dutch Conference on Artificial Intelligence, held on October 23-24, 2003, in Nijmegen. This year's organizers are SNN and University of Nijmegen.

Being the annual conference of the BNVKI (Belgium-Netherlands Association for Artificial Intelligence), BNAIC's aim is to be a meeting place for all AI researchers in Belgium and the Netherlands, and to provide an overview of state-of-the-art AI research and applications in these countries. For these reasons BNAIC traditionally does not only call for original papers, but also for papers that have already been published or accepted for publication elsewhere, as well as for demonstrations of AI systems.

This year 51 original papers have been received. Each paper has been reviewed by two to four referees. 28 papers have been accepted for oral presentation and 20 for poster presentation. All of those are published in the proceedings as full 8-page paper. In addition 33 previously published or accepted papers, and 7 demonstrations have been received. Like previous year, all of those have been accepted for presentation and are published as two page abstracts.

Apart from these contributions, we are very pleased to have this year's invited speakers: Michael Kearns (University of Pennsylvania), John Platt (Microsoft Research, Redmond), and Henk Barendregt (Nijmegen University). In addition, Michael Kearns gave a tutorial on computational game theory, which was offered free of charge with support of SIKS (School voor Informatie en KennisSystemen) and NICI (Nijmegen Institute for Cognition and Information). We hope that this new initiative is appreciated and perhaps even grows into a new BNAIC tradition.

This year the conference is collocated with the workshop *Learning Solutions* on 22 October, 2003. The collocation aims to promote the interaction between researchers in AI and industry. 22 BNAIC papers addressing industrial applications have been invited for poster presentations at the workshop.

We would like to thank our sponsors SIKS, NICI, NWO, SKBS, KUN, Elsevier, SPSS for their financial support. In addition, we would like to thank the members of the program committee for reviewing the papers, the organizers of the workshop on learning solutions for collaboration, the BNVKI for their support and advice, previous year's organizers for sharing experiences and computer codes, and Annet Wanders and webmaster Marco Bloemendaal for their tremendous efforts to make this conference a success.

Nijmegen, October 2003

Wim Wiegerinck
Louis Vuurpijl
Peter Lucas
Tom Heskes

Tutorial

Michael Kearns
"Computational Game Theory"

Invited Speakers

Michael Kearns
"Computational Learning Theory: A Retrospective"

Henk Barendregt
"The Challenge of Computer Mathematics"

John Platt
"Robust Audio Fingerprinting: Feature Selection and Search"

Program Chairs

Tom Heskes
Peter Lucas
Louis Vuurpijl
Wim Wiegerinck

Program Committee

Ameen Abu-Hanna, Pieter Adriaans, Diderik Batens, Cor Bioch, Antal van den Bosch, Herman Bruyninckx, Maurice Bruynooghe, Walter Daelemans, Luc De Raedt, Doug DeGroot, Marc Denecker, Marco Dorigo, Bob Duin, Linda van der Gaag, Pascal Gribomont, Frank van Harmelen, Jaap van den Herik, Wiebe van der Hoek, Eduard Hoenkamp, Jean-Marie Jacquet, Catholijn Jonker, Uzay Kaymak, Etienne Kerre, Joost Kok, Henk Koppelaar, Walter Kosters, Ben Kröse, Han La Poutré, Bernard Manderick, Thierry Massart, John-Jules Meijer, Bart de Moor, Anton Nijholt, Ann Nowe, Niels Peek, Mannes Poel, Eric Postma, Jan Ramon, Jean-Francois Raskin, Gerard Renardel de Lavalette, Silja Renooij, Leon Rothkrantz, Ad Scheepmaker, Lambert Schomaker, Guus Schreiber, Arno Siebes, Maarten van Someren, Ida Sprinkhuizen-Kuyper, Johan Suykens, Yao-Hua Tan, Jan Van den Bussche, Joos Vandewalle, Koen Vanhoof, Frank Veltman, Sofie Verbaeten, Rineke Verbrugge, Floor Verdenius, Nikos Vlassis, Frans Voorbraak, Hans Weigand, Ton Weijters, Marco Wiering, Jef Wijsen, Cilia Witteman, Cees Witteveen, Pierre Wolper.

Contents

Full Papers

Extended Abstracts

Demonstrations

Part 1

Full Papers

Agents, Markets, and Control

Hans Akkermans [‡¶] **Jos Schreinemakers** [‡] **Koen Kok** [§]

[‡]Free University Amsterdam VUA
FEW/I, Business Informatics BI (Email: jos@cs.vu.nl)
[§]Energy Research Centre of the Netherlands ECN
TSC, Petten, The Netherlands (Email: j.kok@ecn.nl)
[¶]AKMC Knowledge Management BV
Klareweid 19, 1831 BV Koedijk, The Netherlands

Abstract. *We present a new and general theory of agent-mediated market-based control. The central result is our derivation of a general market theorem for agent based control that, among others, covers PID control, the most significant class of controllers in practical industrial use. Our market-based control theorem proves two important properties: (1) market-based control is able to handle scarce resources for control in a way that is optimal locally as well as globally ('societally'); (2) in the absence of resource constraints it reduces to a collection of local independent controllers that behave in accordance with conventional control engineering theory. We derive some analytical results for the market outcome, the equilibrium price, and central 'omniscient' control, that help to interpret market-based control in terms of both economic and control theory. Our agent-based market theory can be further generalized to other, more sophisticated, types of control, and it embodies a natural unification of formal microeconomic market and control theory.*

1 Issues and Conjectures in Agent-Based Control

Agent tasks can be categorized into three broad groups: (i) information-oriented tasks; (ii) transaction-oriented tasks; (iii) action-oriented tasks. Personal agents that search and collect information on the (Semantic) Web are an example of (i). Trading agents negotiating on marketplaces and auctions are an example of (ii). Tasks whereby agents directly act in and upon the world go a step further, often requiring subtasks falling within categories (i) and/or (ii). Control is one example of an action-oriented task (iii), and is the subject of the present paper. We will especially focus on market-based control which is a combination of category (ii) and (iii) tasks, whereby a large number of agents are competitively negotiating and trading on a marketplace, with the purpose to optimally achieve their *local* control action goals.

As this is an example of an agent society, an important issue is what predictions one can make about the emergent societal outcome of the local agent interactions, in the present case about the resulting *global* control actions and their optimality. In general, this is an extremely hard and open problem. A lot of work has been done on how to design individual agents and their communication, but results on generally applicable multi-agent systems theory at the 'society level' are much more sparse. In this paper, we

present such a theory of agent-based control, based on an integration of formalisms from microeconomic market theory as well as control theory.

A practically relevant example of agent-based control — we will use it as the running illustration in this paper — is climate control of large buildings with many office rooms. As depicted in Figure 1, offices are attached to a pipe through which cold air is transported and distributed to the individual office rooms. This large-scale control problem has been treated by several authors, and has given rise to different scientific claims regarding the case in favour of agent- and market-based control. Huberman and Clearwater [7, 5] constructed and tested a working multi-agent solution based on a market approach, that they reported to outperform existing conventional control. In their work, Huberman and Clearwater claim decentralized market-based control to be *a better alternative* for conventional control, essentially because centralized 'omniscience' is practically speaking impossible to achieve. Ygge and Akkermans [12] re-analyzed this problem in depth. They showed that other market designs — that strictly respect the agent principles of local information and action — work even better. Also, however, they constructed a central conventional control engineering solution that yields equivalent results. These authors view market-based control not as a better alternative, but rather as a decentralized *equivalent* formulation of central conventional control. This leads them to the qualitative conclusion that *"local information + market communication = global control"*. This outcome equivalence of market-based control and central conventional control was stated as a conjecture because it was based on a specific application with a specific type of conventional control.

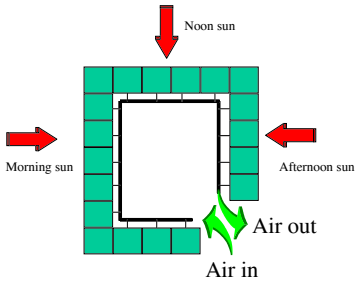


Figure 1: *A multi-agent control problem for smart self-managing office buildings.*

In this paper we will deal with these issues, and answer several of the open questions. The central result is our derivation (Sec. 2) of a general equilibrium market theorem for agent-based control. This market-based control theorem proves two important properties: (1) market-based control is able to handle scarce resources for control in a both locally and globally ('societally') optimal way; (2) in the absence of resource constraints it reduces to a collection of local independent controllers that behave in accordance with conventional control engineering theory. We mention in passing that, although the focus of the present paper is theoretical, market-based concepts as developed here have been practically implemented and tested in simulation studies as well as real-world field experiments concerning comfort management in intelligent office buildings by means of *'HomeBot'* agents (see [3, 10, 1, 9]).

2 Market Theory and Protocol of Global PID Control

PID control. In this paper we develop the market-based control theory for one significant class of local controllers called PID control. PID stands for Proportional-Integral-Derivative control, and the reason for this name is formally explained in its control rule equation below. PID control is a simple and standard tool to solve control problems [2]. Although it is quite old and there are many other, more advanced forms of control (which we will consider in forthcoming papers), PID control is still widely used and dominant in industrial practice. The building control problem described above is a case in point: commercial building and energy management systems include a large number of local PID controllers. The general equation expressing the PID control strategy is

$$r(t) = K^P x(t) + K^I \int_0^t dt' x(t') + K^D \frac{d}{dt} x(t). \tag{1}$$

Here, $r(t)$ is the 'input' resource variable, $x(t)$ is the 'output' state error variable, the difference between the actual state and the goal state (setpoint), and the K are called the gain constants. The first term with K^P represents the proportional control strategy, the second term with K^I represents an integral controller, and the third term with K^D gives the derivative control part (hence the name PID control). It is assumed throughout that all variables, parameters and functions are real. In market terms, $r \in \Re$ implies that we are dealing with an infinitely divisible resource. In the previously sketched application of climate control of large office buildings x stands for the error in room temperature and r is the heating or cooling power. Agent studies of market-based control of smart buildings typically used a special case of Eq. (1), viz. integral controllers ($K^P = K^D = 0$).

Now, let us consider a large collection $\alpha = 1, \ldots, N$ of independent local PID controllers (in the running application example, one local controller for each room in the office building; so N may run in the hundreds or even thousands). Each local controller follows the PID control rule of Eq. (1), which can be more concisely written in operator notation as $r_\alpha(t) = \mathcal{O}_\alpha(t) x_\alpha(t)$, where $\mathcal{O}_\alpha(t) \stackrel{\text{def}}{=} K^P_\alpha + K^I_\alpha \int_0^t dt' + K^D_\alpha \frac{d}{dt}$ is a *linear operator* operating on the local state or error variable $x_\alpha(t)$. We note that the above equation is identical to Eq. (1); it is only written in a different notation that is more compact and convenient for the theorems and derivations later in the paper. It also shows that PID control is a function of the current time instant t, and thus is a form of *instantaneous* control. As a matter of convenience, in what follows we will no longer indicate the dependence on t explicitly, so we will write this simply as $r_\alpha = \mathcal{O}_\alpha x_\alpha$. Notably, $r_\alpha = \mathcal{O}_\alpha x_\alpha$, the shorthand for Eq. (1) provides the full generalization of various special cases investigated by previous agent control studies to general PID control strategies deployed by individual agents.

The conceptual interpretation that we now introduce in the context of agent-based control, is that the PID control rule Eq. (1) states the resource amount r_α that a controller agent α needs if it wants to achieve its goal state (setpoint) by eliminating the state error x_α. The PID control rule thus defines the agent's local demand function without taking possible resource constraints into account. If the total resource is constrained, it is traded on a marketplace that thus serves as a resource allocation mechanism. This scarce-resource situation is not addressed in conventional control, and this is the added value of the market approach. This is formalized below.

General market theorem for agent-based control. We are now set to study market-based large-scale control. We assume an agent society in which each individual agent α represents a local independent controller, and has to negotiate with other agents in a marketplace in order to obtain its desired resource r_α. In particular, there may occur a situation of scarce resources, in the sense that the total available resource at a certain time t is smaller than the sum total of the requested resources $\sum_{\alpha=1}^{N} r_\alpha$ as given by Eqs. (1) . In this new constrained situation, how are the available resources going to be distributed, what is the optimal situation for each agent locally, how does its new local control strategy look like, and what is the resulting global control strategy in this agent society?

Definition 2.1 *Let each independent local PID controller be represented by an agent α, whereby its utility function is defined by*

$$u_\alpha(r_\alpha) = f_\alpha(r_\alpha), \tag{2}$$

where f_α is a strictly concave function of r_α that is twice continuously differentiable (on a suitable interval $|r_\alpha| \leq |R^{unc}|$), and that has its maximum at the local resource value $r_\alpha = \mathcal{O}_\alpha x_\alpha$ as given by the PID control equations Eq. (1). Furthermore, suppose that the total available resource is scarce, so that we have

$$0 \leq \sum_{\alpha=1}^{N} r_\alpha = R^{max} \leq \sum_{\alpha=1}^{N} \mathcal{O}_\alpha x_\alpha \stackrel{def}{=} R^{unc}. \tag{3}$$

Finally, let all agents be self-interested utility maximizers and let them be competitive. [1]

The latter equation says that R^{unc} is the total 'free' demand of all agents taken independently, as implied by the sum total of the PID control equations in a situation with an unconstrained supply of resources. But, there may be a smaller cap R^{max} on the total available resource if it has to be shared by the agent society (as is the case with the cooling or heating power in the smart building example). The total demand R^{unc} by the individual agents in the unconstrained case derives from local information , whereas the resource limitation to R^{max} is the result of a supposed external action or situation. One has the design freedom to choose *any* agent utility function f_α within the confines of Definition 2.1.

Theorem 2.1 *Assuming the agent utility functions and behaviours given by Definition 2.1, the following statements hold:*

A *There exists a resource allocation r_α^* that is a global maximum to the optimization problem:* $\mathrm{Max} \sum_{\alpha=1}^{N} f_\alpha(r_\alpha)$ *subject to the resource constraint $\sum_{\alpha=1}^{N} r_\alpha = R^{max}$, and this global maximum is unique.*

B *The* same *resource allocation r_α^* is* identical *to the competitive equilibrium of a market in which each individual agent maximizes its utility $f_\alpha(r_\alpha)$, subject to $p^* \cdot r_\alpha^* \leq p^* \cdot R^{max}$, whereby the market is clearing (i.e. $\sum_{\alpha=1}^{N} r_\alpha^* = R^{max}$) and p^* is the market clearing price.*

[1] Competitiveness in microeconomic theory means that agents take prices as externally given so that they are not able to directly influence or set the market clearing price (as is the case in a monopoly or oligopoly situation).

C *The resource allocation r^*_α obtained as outcome of this competitive equilibrium market is Pareto optimal (i.e. there exists no other allocation that is better or neutral for* all *agents).*

Proof. Statement A follows directly from Definition 2.1 and standard microeconomic and optimization theory (see for example Mas-Colell et al. [11], pp. 50-51, 314-327, 945, 962, Ibaraki and Katoh [8], Chapter 2, and Fletcher [6], pp. 216-218). A standard tool to obtain the constrained maximizer in an optimization problem is to set up the Lagrangian $\mathcal{L}$ for problem A, as follows : $\mathcal{L} = \sum_{\alpha=1}^{N} f_\alpha(r_\alpha) + p\left(R^{max} - \sum_{\alpha=1}^{N} r_\alpha\right)$, and then to solve the equation system $\frac{\partial \mathcal{L}}{\partial r_\alpha} = 0$ and $\frac{\partial \mathcal{L}}{\partial p} = 0$. The latter equation immediately yields the constraint equation $\sum_\alpha r_\alpha = R^{max}$. The former equation gives $\frac{\partial f_\alpha}{\partial r_\alpha} = p$ for all α: the marginal agent utilities all have the same value p at equilibrium. The resource allocation r^*_α that satisfies these equations solves the *(global)* optimization problem of statement A. Now, the definition of competitive equilibrium (e.g. [11], p. 314) implies that the *(local)* Lagrangian L_α for each individual agent in the market problem in statement B is: $L_\alpha = f_\alpha(r_\alpha) + p\left(R^{max} - \sum_{\alpha=1}^{N} r_\alpha\right) + q(p^* \cdot R^{max} - p^* \cdot r_\alpha)$. Considering the equation $\frac{\partial L_\alpha}{\partial r_\alpha} = 0$ that holds at market equilibrium, we observe that the *same* equation obtains for the market problem B as for the optimization problem A. Thus, the resource allocation r^*_α is the solution to the competitive equilibrium market problem of statement B. This proves statement B. Statement C now immediately follows from the first fundamental theorem of welfare economics ([11], pp. 325-327). □

Top-down central control or bottom-up decentralized markets? Our market theorem for agent-based control is the key contribution of this paper. It has weaker assumptions and is much stronger than the corresponding result in [12]. It also holds in the *multicommodity* case. In fact, it is a rather generic statement about the relationship between global optimization and competitive market problems. Consequently, it is more generally applicable, also beyond PID control. The optimization problem of statement A reflects how a central controller, that oversees all local control agents, will look at the situation in a top-down fashion. In contrast, the market problem of statement B represents the situation through the eyes of the individual agents in a bottom-up manner. Our market theorem then shows that there is an *outcome equivalence* between the two approaches. Moreover, statement C says that the resulting resource allocation r^*_α is optimal both locally and globally, in other words, it is optimal for each agent individually as well at the agent society level.

Market protocols for agent-based PID control. Any suitable so-called resource-oriented market protocol will do, for example a (quasi)Newton-type algorithm, as discussed in detail by Ygge and Akkermans [13]. In this case, we search for the equilibrium resource allocation r^*_α, which is updated in each market round by the auctioneer agent; each control agent essentially submits its marginal utility $\frac{\partial f_\alpha}{\partial r_\alpha}$ which may be interpreted as its *bid price*. Alternatively, price-oriented market protocols are also possible, for example Wellman's WALRAS [4]. Then, we search for the equilibrium value of p, whereby p is interpreted as the *going market price* which is updated in each bidding round by the auctioneer agent, and the control agents submit their demand r_α. The price-oriented protocol requires an explicit equation for the agents' demand function. Explicit expressions for the demand functions of PID control agents will be derived in the next section.

3 Analytical Results in Market-Based Control

The general statements of the previous section can be extended to analytical results that throw further light on the characteristics of market-based PID control if we consider some special cases. Namely, the simplest utility function that satisfies the requirements of Theorem 2.1 is a quadratic one: [2] $u_\alpha^Q(r_\alpha) = -\frac{1}{2c_\alpha}(r_\alpha - \mathcal{O}_\alpha x_\alpha)^2$, where $w_\alpha \stackrel{\text{def}}{=} 1/c_\alpha$ is a weight factor > 0 that may be different for each agent. The weight factor may be used to express individual preference differences (a higher w makes the utility function sharper and a lower one makes it broader, so that the agent less/more easily makes concessions in its utility maximization) and/or to express some social hierarchy (the boardroom is probably seen as more important to serve than the junior underassistant's office); c can be seen as a measure for the agent's willingness to make concessions. The differences in strictly physical characteristics (such as room size) are already catered for through the $\mathcal{O}_\alpha x_\alpha$ term in the utility equation.

This form of the utility function allows us to analytically solve the Lagrangian equations for the constrained market optimization:

$$\frac{\partial \mathcal{L}_Q}{\partial r_\alpha} = 0 \;\Rightarrow\; r_\alpha = \mathcal{O}_\alpha x_\alpha - c_\alpha \cdot p_Q \tag{4}$$

Through this equation we have actually derived an explicit expression for the agent's demand function when available resources are scarce . Using the resource constraint equation together with Eq. (3) then gives the market outcome at equilibrium:

$$p_Q^* = \frac{(R^{unc} - R^{max})}{\sum_{\alpha=1}^N c_\alpha}, \text{hence } r_\alpha^* = \mathcal{O}_\alpha x_\alpha - \frac{c_\alpha}{\sum_{\alpha=1}^N c_\alpha}(R^{unc} - R^{max}). \tag{5}$$

The latter equation is actually a generalized version of the central control solution constructed by [12]. The above analytical results are useful in their own right, because we have the agent design freedom to choose the utility function. Hence, it is natural to take the simplest one (viz. the quadratic form) that does the job.

4 Discussion: The Microeconomics of Control

By looking at the proof of Theorem 2.1 and the subsequent discussion on market protocols in Sec. 2, it follows that the Lagrangian multiplier p is naturally interpreted as the going market price. Hence, its value at the optimal resource allocation r_α^* that represents the competitive market equilibrium is equal to the market clearing price p^*. If we apply this interpretation to the analytical results in Sec. 3, we see that the market clearing price for constrained PID control is proportional to $(R^{unc} - R^{max})$, in other words, to the 'cut' applied to or shortage in the total available resource. In the unconstrained case, the equilibrium price is zero. Thus, the unconstrained situation of independent local controllers is the limiting case of the constrained control problem, whereby there is no resource cut.

[2] As a practical remark, we note that simple utility functions like these *do* make sense in real-life applications such as smart buildings, see e.g. [12, 3, 9].

Decentralized markets versus central 'omniscient' control. Let us reconsider Equation (5). It has the form of a PID control rule extended with a term reflecting resource constraints. By summing it over α it is easily seen that it incorporates the resource constraint (both sides always sum to R^{max}). Moreover, it also covers the unconstrained case (because then $R^{max} = R^{unc}$, so that the term with the weight factors w equals zero, and we are left with Eq. (1)).

Hence, Equation (5) actually gives the recipe for the design of a *central* 'omniscient' PID controller that behaves under resource constraints in a way *equivalent* to the *decentralized market* approach to PID control. These equations show that the central controller needs global access to the following information: (1) all local PID control rule information (i.e. all $\mathcal{O}_\alpha x_\alpha$, in order to be able to construct R^{unc}); (2) all local weights w_α; (3) the value of R^{max}. This formalizes what Huberman and Clearwater's 'omniscience' means. The central controller then can correctly instruct each local controller with the proper (changed) PID rule when the total resource is constrained, and can simply leave them alone in the unconstrained case. This extends the previous partial results by [12] to general PID control.

We note that the direct explicit construction of the central controller fully depends on whether an analytical solution of the constrained optimization equations can be found. This is possible due only to the special choice of a quadratic utility. It is *not* straightforward or even possible in the general case, whereas the decentralized market approach will *always* work properly (according to Sec. 2). This completes the formal generalization of market-based control to large-scale PID control. It also settles some conjectures and outstanding questions regarding the capabilities and added value of market-based control versus central conventional control. In brief, we have demonstrated:

- For resource-constrained large-scale PID control, there always exists a central conventional controller (Sec. 2, market theorem statement A).
- There also always exists a Pareto-optimal market solution, in outcome identical to the mentioned central controller (Sec. 2, market theorem statement B and C).
- The computational market can always be constructed in the general PID case, and suitable market protocols have been given (Sec. 2).
- In contrast, a direct explicit construction of the central controller is only possible in a few special cases (Sec. 3). In the general case, a computational market approach is needed (or at least a functionally equivalent optimization algorithm).
- The central controller and the market-based control solution both control the local PID controllers, but in the general case neither of them is *itself* a PID controller. This is only true in the special case of quadratic utility functions and linear demand functions of the control agents (Sec. 3).

Market-based control as an online adaptive strategy. In essence, market-based control is a (very) special type of what is known as *adaptive* control. It embodies an, online and self-organizing, adaptation of the local control strategies when overall resource limitations come into play. The results of this paper have been derived for the general case of PID control. They can, however, be readily generalized for other, more sophisticated types of control, for example state feedback control. This will be demonstrated in forthcoming papers. State feedback handles the multi-input multi-output case with multiple resources and state variables. All results of the present paper carry over to state feedback

control, whereby the market is turned into a multicommodity market. Our market-based approach is also applicable for example to so-called quadratic optimal control: this optimizes a chosen performance index over a whole time period (so it is not simply a form of instantaneous control as PID or state feedback are). In that case, the market bidding rounds are inherently intertwined with internal iterative optimization computations by each local control agent, which is responsible for a (possibly large) multidimensional dynamic subsystem. A direct central control approach then ceases to be possible at all, whereas market-based control offers an effective as well as conceptually natural decentralized computational model. In sum, our market-based control approach yields suitable formalisms — that integrate microeconomic and control theories — and associated computational mechanisms that obtain global optimal control in an online adaptive way when needed resources are scarce.

Acknowledgments. This work was partially supported by EnerSearch AB (Malmö, Sweden) and by the European Commission in the context of the EU-EESD Project No. NNE5-2001-00906 called CRISP (Distributed Intelligence in CRitical Infrastructures for Sustainable Power).

References

[1] J.M. Akkermans. Being smart in information processing — technological and social challenges and opportunities. In M. Musen, B. Neumann, and R. Studer, editors, *Intelligent Information Processing*, pages 1–12, Norwell, MA, 2002. Kluwer Academic Publishers. Keynote IIP-IFIP World Computer Congress, Montreal, August 2002.

[2] K-J. Åström and B. Wittenmark. *Computer-Controlled Systems - Theory and Design*. Prentice Hall, Englewood Cliffs, NJ, 1990. ISBN 0-13-168600-3.

[3] E. Boertjes, J.M. Akkermans, R. Gustavsson, and R. Kamphuis. Agents to achieve customer satisfaction — the COMFY comfort management system. In *Proceedings of the Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM2000)*, pages 75–94, Blackpool, UK, April 2000. The Practical Application Company Ltd.

[4] J. Cheng and M.P. Wellman. The WALRAS algorithm—a convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 12:1–24, 1998.

[5] S.H. Clearwater (Ed.). *Market-Based Control — A Paradigm for Distributed Resource Allocation*. World Scientific, Singapore, 1996.

[6] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987. Second Edition.

[7] B.A. Huberman and S.H. Clearwater. A multi-agent system for controlling building environments. In V. Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 171–176, Menlo Park, CA, June 1995. AAAI Press / The MIT Press.

[8] T. Ibaraki and N. Katoh. *Resource Allocation Problems—Algorithmic Approaches*. The MIT Press, Cambridge, MA, 1988.

[9] J. Jelsma, A. Kets, R. Kamphuis, and W. Wortel. SMART field test: Experience of users and technical aspects. Technical Report ECN-C–02-094, Energy Research Centre of The Netherlands ECN, Petten, The Netherlands, October 2002.

[10] R. Kamphuis, C. Warmer, W. Zeiler, W. Wortel, J.M. Akkermans, and J. Jelsma. SMART — experiences with e-services for smart buildings. In F.-N. Pavlidou, editor, *Proc. IEEE 6th Int. Symp. on Power-Line Communication and its Applications (ISPLC-2002, Athens, 27-29 March 2002)*, Thessaloniki, Greece, 2002. Aristotle University of Thessaloniki and IEEE. CD-ROM available from IEEE.

[11] A. Mas-Colell, M. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[12] F. Ygge and J.M. Akkermans. Decentralized markets versus central control: A comparative study. *Journal of Artificial Intelligence Research*, 11:301–333, October 1999. Available from http://www.jair.org.

[13] F. Ygge and J.M. Akkermans. Resource-oriented multicommodity market algorithms. *Autonomous Agents and Multi-Agent Systems*, 3(1):53–71, 2000. AAMAS Journal Special Issue Best Papers of ICMAS'98.

Application of Cluster Variation Method to Genetic Linkage Analysis

Kees Albers [a] Bert Kappen [a]

[a]University of Nijmegen, Department of Biophysics, Geert Grooteplein 21, 6525EZ Nijmegen, The Netherlands

Abstract

In this paper we discuss the application of the Cluster Variation Method to the problem of genetic linkage analysis. The objective of genetic linkage analysis is to link an observed affection status (phenotype) of all individuals in a pedigree to the inheritance pattern observed certain locations on the chromosome. The Bayesian approach introduced by [6] is computationally intractable for large pedigrees and we propose to apply the Cluster Variation Method to approximately compute quantities such as the likelihood of the data. We obtain good results on an important subproblem, the computation of the intractable probability distribution of the inheritance pattern. These results encourage the incorporation of the disease model into the inheritance model.

1 Genetic Linkage Analysis

One of the major questions in genetics is to link an observed disease to a certain region on the chromosome. When a disease is thought to be (partially) genetic, scientists collect a large data base of pedigree data from families in which usually a few affected individuals occur. Most or all of the family members are genotyped with markers, a technique which can be used to reconstruct the way genetic information has passed along the generations (very similar to tests for fathership).

The objective of genetic linkage analysis is to link an observed affection status (phenotype) of all individuals in a pedigree to the inheritance pattern observed certain locations on the chromosome. The affection status can be directly observed for each individual in the pedigree. The disease model that determines the relation between affection status and genotype may be unknown as well as the number of locations on the chromosome that cause the disease. Through DNA-analysis there is information about the inheritance pattern, i.e. information about which genes were passed on from parents to children.

For reconstructing the inheritance pattern in a pedigree, there exist algorithms, which are exact and make use of all available marker data. They are either exponential in the number of locations on the chromosome taken into account [1] or the size of the pedigree [5].

For each locus on the chromosome, the inheritance pattern within a pedigree is denoted by the inheritance vector v_i, which contains two binary valued variables for each non-founder[1]. The subscript i denotes the location at the chromosomes. The binary vari-

[1]Non-founders are individuals in the pedigree whose parents are also in the pedigree.

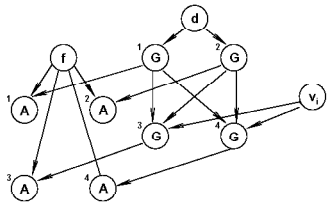


Figure 1: The graphical model to link a genetic disease to a certain inheritance pattern for a very small pedigree of four individuals. The nodes labeled with 'A' are affection states (affected or unaffected), which are directly caused by 'G', the genotype of the individual and 'f', the disease model. The genotype of the founders (the parents or individual 1 and 2) is determined by population statistics 'd', whereas the genotype of the children (individual 3 and 4) can be determined from that of their parents given 'v_i'. This 'v_i', also known as the inheritance vector, consists of four binary states which determine which parental genes are copied.

ables indicate which one of the chromosomal pair is copied from the father and which one is copied from the mother. Unfortunately, the inheritance vector cannot be obtained directly, but an indrect measurement using 'markers' (genotyping) is used to reconstruct v_i. The measured marker data D consists of short identifiable sequences of DNA that can have one out of r values, where r is typically in the range of 2-10. Since in most cases v_i cannot be determined uniquely, one computes a probability distribution $p(v_i|D)$ over all possible v_i.

We will treat linkage analysis as a Bayesian inference problem. In [6] it is shown that using the likelihood of the affection status A given v_i denoted by $p(A|v_i)$ as a scoring function, has significantly more statistical power than existing non-Bayesian methods.

In figure 1 we show the graphical model as proposed in [6]. The figure shows the model belonging to a very small pedigree, but is straightforward to extend it to larger pedigrees. The node 'd' determines the probability that a mutated gene[2] occurs in the population. The probability that an individual has precisely one mutated gene, for instance, is then given by $2d(1-d)$. Once the genotype 'G' of the founders is the pedigree is set, the genotype of the non-founders is completely determined by 'v_i'.

The genotype, G, can be either '–', '-+','+-' or '++', where a plus denotes the presence of a mutated gene. Since there is no biological difference between '-+' and '+-' we only need the probability of the affection status given the number of mutated genes an individual has. For instance, $p(A|\#\text{mutatedgenes}) = \{0, 0, 1\}$ for a purely recessive disease. In principle all probability tables are possible. The exact three values are known as 'penetrance values' and are represented in the figure with the symbol 'f'. It is immediately clear from the graphical model, that (although the penetrance values are often not known) all individuals share the same disease model.

The total score, i.e. the likelihood of the observed affection status for a locus i, is now

[2]The genotype, G consists of the copies of a single gene. Depending on the disease model, one or more of these combinations can cause the disease.

given by

$$S_i = p \int_{v_i} p(A|v_i)p(v_i|D) \tag{1}$$

Once the genotypes of the founders are known, the genotypes of all non-founders can be reconstructed with the inheritance vector v_i. The disease model determines how A depends on the genotype G. Therefore

$$p(A|v_i) = \int_d \int_f \int_G p(A|Gf)p(G|v_i d)p(f)p(d) \tag{2}$$

where f is the disease model and d population statistics. $p(f)$ and $p(d)$ are the respective priors. For a more detailed treatment of the graphical model concerning $p(A|v_i)$ we refer to [6].

The computation of the integral over all states v_i, as well as the individual terms $p(A|v_i)$ and $p(v_i|D)$ in Eq. 1 are intractable when both the number of individuals in the family tree is large and the number of marker loci is large. Therefore approximations are required. In this paper we will focus on the important subproblem of computing marginals of the distribution $p(v_i|D)$ as well as the normalization $p(D)$ using the Cluster Variation Method. In the next section we will discuss in more detail the model for the computation of these quantities.

Efficient treatment of this subproblem is clearly a necessary requirement for efficient approximation of the full problem, and probably sufficient. The reason that we do not treat the full problem is that the form of the disease model $p(A|v)$ is not known (dominant, recessive or other) resulting in the additional complication of integrating over disease models.

2 A probabilistic inheritance model

In this section we will describe the model to compute $p(v_i|D)$. So far we have only considered the disease locus: the location on the chromosome which segregates the mutated gene. In order to determine how it is segregated through the pedigree, information on the inheritance vector v_i on the disease locus is required.

It cannot be determined by direct measurement because the location of the disease locus on the chromosome is unknown and the inheritance pattern of a locus on the chromosome varies with the location of the locus on the chromosome. However it is possible to determine the distribution over inheritance patterns on the disease locus. At fixed locations on the chromosome markers are placed, that yield for each individual and each marker locus two marker values. These marker values correspond to the alleles on that locus of the chromosome. By considering the marker values of all individuals on a locus jointly, it is possible to infer which alleles were transmitted from parents to child using the familial relations in the pedigree. Thus the probability distributions over inheritance patterns on the marker loci can be reconstructed. On the locations between the markers the probability distribution can be inferred by interpolation according to the physical correlations between loci. Since the marker data of a single locus is usually unsufficient to determine the inheritance pattern on the marker locus, we can exploit the same correlation

of the inheritance patterns between the loci to determine the inheritance patterns on the marker loci more accurately.

This leads to an inheritance model which has two components:

1. The first concerns the reconstruction of v_i on marker locus i. For each triplet of parents $\pi(\mu)$ and child μ in the pedigree on locus i, there is a factor $\psi(G^i_\mu, G^i_{\pi(\mu)}, v^i_\mu)$, where G^i_μ represents the genotype of individual μ on locus i. This is a table that has a one for each genotype $(G^i_\mu, G^i_{\pi(\mu)})$ that is compatible with the marker data D.

2. The second concerns the coupling between loci. The loci are coupled via the inheritance bits v^i_μ of each individual. Since these specify a gene is inherited from the father or the mother they are binary. If two loci are close together, the inheritance bits of an individual are more likely to be equal than if the loci are far apart. The probability θ of a change is given by $\theta = \frac{1-e^{-2\alpha}}{2}$, where α is the genetic distance measured in Morgans. This gives the factors

$$\psi(v^i_\mu, v^{i+1}_\mu)$$

These factors define a graphical model of which the joint distribution is given by

$$p(\{G^i_\mu, v^i_\mu\}|D) = \frac{1}{Z} \prod_\mu \prod_i \psi(G^i_\mu, G^i_{\pi(\mu)}, v^i_\mu) \prod_{i=1,\ldots,i=L-1} \psi(v^i_\mu, v^{i+1}_\mu) \tag{3}$$

where Z is the normalization constant and L is the number of marker loci. In order to get the expression required for Eq. 1 we marginalize over the genotypes G^i_μ:

$$p(\{v^i_\mu\}|D) = \sum_{\{G^i_\mu\}} p(\{G^i_\mu, v^i_\mu\}|D)$$

3 Cluster Variation Method

In general $p(\{\phi^\mu_i, v^{\lambda\mu}_i\}|D)$ from Eq. 3 is intractable and therefore approximations are required. The Cluster Variation Method has been developed in the physics community to approximately compute the properties of the Ising model [4, 3].

Let $x = (x_1, \ldots, x_n)$ be a set of variables, where each x_i can take a finite number of values. We can write any positive distribution on x in the following form

$$p_H(x) = \frac{1}{Z(H)} e^{-H(x)} \qquad Z = \sum_x e^{-H(x)} \tag{4}$$

p_H can be obtained as the minimum of the Gibbs free energy, which is a functional over probability distributions of the following form:

$$F_H(p) = \langle H \rangle + \langle \log p \rangle \,, \tag{5}$$

where the expectation value is taken with respect to the distribution p, i.e. $\langle H \rangle = \sum_x p(x) H(x)$. When one minimizes $F_H(p)$ with respect to p under the constraint of normalization $\sum_x p(x) = 1$, one obtains p_H.

The Cluster Variation Method replaces the probability distribution $p_H(x)$ by a set of (possibly overlapping) probability distributions B, each describing the interaction between a small number of variables. Each cluster α determines a probability distribution $p_\alpha(x_\alpha)$; together the clusters $\alpha \in B$ determine the approximation. The Cluster Variation Method defines an approximate free energy in terms of the clusters $\alpha \in U$, where U is the set of all clusters $\alpha \in B$, intersections of clusters $\alpha \in B$ and intersections of intersections of clusters, etc. This free energy must be minimized subject to normalization constraints $\sum_{x_\alpha} p_\alpha(x_\alpha) = 1$ and consistency constraints

$$\sum_{x_\alpha \backslash x_\beta} p_\alpha(x_\alpha) = p_\alpha(x_\beta) = p_\beta(x_\beta), \quad \alpha, \beta \in U,\ \beta \subset \alpha. \tag{6}$$

We enforce the constraints by adding Lagrange multipliers to the free energy, resulting in the cluster variation free energy:

$$F_{\mathrm{cvm}}(\{p_\alpha(x_\alpha)\}, \{\lambda_\alpha\}, \{\lambda_{\alpha\beta}(x_\beta)\}) = \sum_{\alpha \in U} a_\alpha \sum_{x_\alpha} p_\alpha(x_\alpha) \log \frac{p_\alpha(x_\alpha)}{\psi_\alpha(x_\alpha)}$$
$$- \sum_{\alpha \in U} \lambda_\alpha \left(\sum_{x_\alpha} p_\alpha(x_\alpha) - 1 \right) - \sum_{\alpha \in U} \sum_{\beta \subset \alpha} \sum_{x_\beta} \lambda_{\alpha\beta}(x_\beta) \left(p_\alpha(x_\beta) - p_\beta(x_\beta) \right) \tag{7}$$

The Moebius numbers a_α are defined by

$$1 = \sum_{\alpha \in U, \alpha \supset \beta} a_\alpha, \quad \forall \beta \in U$$

and correct for overcounting of the free energies of the clusters $\alpha \in B$. For more details on this approximation, we refer to [3].

One can run Generalized Loopy belief propagation on this problem [7], but there is no guarantee that this algorithm converges. In fact, on this particular problem it does not converge. A provably convergent algorithm is the CCCP method proposed by Yuille [8], which requires a double loop approach, where at each outer loop iteration a convex subproblem with linear constraints is defined, which is solved using standard methods. In [2] this double loop method was significantly improved, essentially by defining an improved convex subproblem at each outer loop iteration. We refer to [2] on further details of this method.

4 Likelihood of the data

The CVM method can also be used to approximate the normalization constant of the probability distribution, which in the presence of evidence is equal to the likelihood of the evidence. This is the quantity that is computed in Eq. 1 for disease models that consider each locus (or subset of loci) in turn as the cause of the disease. Here we show that the CVM method can be used effectively to approximate the likelihood of the data D.

We have

$$p(v, G|D) = \frac{p(v, G, D)}{p(D)}$$

Defining

$$\psi_D(v, G) \equiv p(v, G, D) \quad \text{and} \quad Z_D = p(D)$$

we obtain a distribution of the form 4. The minimum of the free energy equals $-\log Z$ and thus here relates to the data likelihood $p(D)$. The minimum of the approximate free energy F_{cvm} therefore gives an approximation of the data likelihood $p(D)$.

5 Methods

The goal of the simulations is to determine the quality of the CVM approximation of the likelihood of the data $p(D)$ and marginals of $p(v_i|D)$. This requires the exact distribution to be tractable.

We will use a pedigree of 28 individuals of which 10 are founders and 18 are non-founders. All parents have two children. The graphical model corresponding to this pedigree is shown in figure 5. There is no inbreeding, so for a given locus the corresponding graphical model is a tree. Therefore we can compute the exact distribution for networks consisting of up to 4 loci given this pedigree, by collapsing the 4 loci into a single locus tree-structured graph and running the junction tree algorithm on this graph.

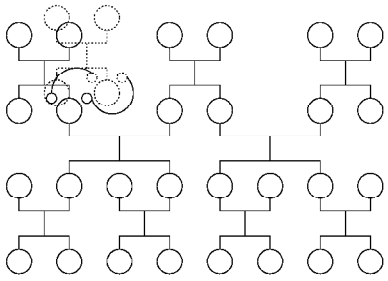


Figure 2: The graphical model corresponding to the pedigree. The large nodes indicate genotypes of the individuals. Small nodes indicate the inheritance bits of each individual. The dashed nodes correspond to genotypes and inheritance bits on the adjacent locus.

In the simulation study we investigate the effect of the parameters α and $|m|$ on the quality of the approximation. α is the genetic distance between the marker loci and determines the strength of the correlation between the loci. It is expected that the approximation becomes more accurate as the coupling between the loci becomes weaker. $|m|$ is the number of marker values and in part determines the accuracy with which the inheritance pattern can be reconstructed. Here we will consider $\alpha = \{0, \ldots, 80\}$ and $|m| = 5$, which give results that are representative of other values of $|m|$. With these parameters a random inheritance pattern is generated according to α, and corresponding to the inheritance pattern a set of compatible marker values is generated according to $|m|$. For a given set of $(\alpha, |m|)$ we generate 10 networks.

For the CVM-approximation we have to make a choice for the set of basic clusters B. The guideline is to have the cluster encompass the strongest correlations and to resolve the smallest loops. A requirement is that for any pedigree and number of loci the clusters

must be tractable. For instance a cluster choice depending on the number of children of two parents is therefore unfeasible for pedigrees with many children.
We will consider two cluster choices:

- C_1: Clusters are defined according to the potentials ψ.
- C_2: Clusters consist of parent-child nodes of adjacent loci:
  B=$\{G^i_\mu, G^i_{\pi(\mu)}, v^i_\mu, G^{i+1}_\mu, G^{i+1}_{\pi(\mu)}, v^{i+1}_\mu\}$.

Since the CVM free energy may exhibit local minima, for each problem we run the double-loop algorithm of [2] 5 times with random initialization of the cluster probabilities to detect local minima. The algorithm is considered converged if the maximum absolute change in the cluster probabilities Δp is smaller than $1 \cdot 10^{-6}$.

6 Results

Figure 6 shows the results for clusters C_1 and $L = 4$ loci. The upper two plots show the maximum MAD (maximum absolute error) of all cluster marginals in the network for each network. The left figure shows the MAD for networks for which F_{cvm} has multiple minima, the right figure shows the MAD for networks for which F_{cvm} has a single minimum, meaning that the algorithm converged to the same solution in five restarts. We see that the presence of multiple minima is related to large errors. Also for small α the error increases due to the stronger correlations between the marker loci. Enlarging the clusters removes many of the large errors and multiple minima; for clusters C_2 we find that MAD < 0.1 for $\alpha > 12$. We also find that error increases with the number of loci.

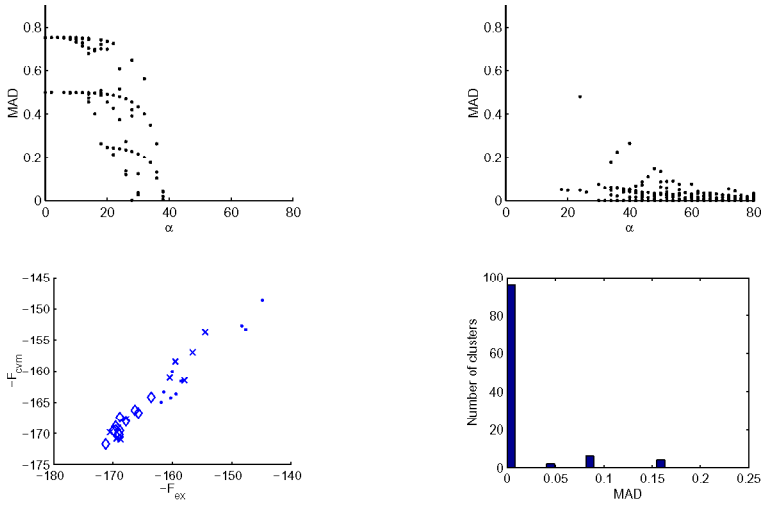


Figure 3: Clusters C_1 and $L = 4$. Starting from upper left clockwise: MAD vs α for networks with multiple minima; MAD vs α for networks without multiple minima; approximation of log-likelihood $-F_{\mathrm{CVM}}$; histogram of errors for $\alpha = 12$

In the bottom-right figure we have plotted the typical histogram of the MAD's of all cluster marginals in the network for a network of $\alpha = 12$ and clusters C_1. The error is large only for a small number of clusters in the network. The consequence is that the approximation of the log-likelihood is still fairly accurate. This is shown in the bottom-left figure. Here we compare the approximated log-likelihood given by F_{cvm} to the exact log-likelihood computed with the junction tree algorithm for $L = 4$ and C_1. For C_1 we find that the maximum error of the approximate log-likelihood is 4 %, for clusters C_2 the maximum error is 1 %.

We conclude that even with the small clusters C_1 the approximated marginals log-likelihood are fairly accurate and that enlarging the clusters further reduces error.

7 Conclusions

We conclude that the proposed method gives accurate approximations of the likelihood of the data and marginals of the inheritance model on tractable networks. Since the addition of the disease locus is very similar to adding another marker locus, we argue that the Cluster Variation Method is a very promising approximation method in the context of genetic linkage analysis.

References

[1] R.C. Elston and J. Stewart. A general model for the genetic analysis of pedigree data. *Human Heridity*, 21:523–542, 1971.

[2] T. Heskes, K. Albers, and H.J. Kappen. Approximate inference and constrained optimization. *UAI-2003*.

[3] H. J. Kappen and W. Wiegerinck. Novel iteration schemes for the cluster variation method. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 415–422, Cambridge, MA, 2002. MIT Press.

[4] R. Kikuchi. *Physical Review*, 81:988, 1951.

[5] L. Kruglyak, M.J. Daly, M.P. Reeve-Daly, and E.S. Lander. Parametric and nonparametric linkage analysis:a unified multipoints approach. *American Journal of Human Genetics*, 58:1347–1363, 1996.

[6] M.A.R. Leisink, H.J. Kappen, and H. Brunner. Linkage analysis: A bayesian approach. In *Proceedings ICANN*, 2002.

[7] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13 (Proceedings of the 2000 Conference)*, 2001. In press.

[8] A.L. Yuille. CCCP algorithms to minimize the bethe and kikuchi free energies: Convergent alternatives to belief propagation. *Neural computation*, 14:1691–1722, 2002.

A New Exam Timetabling Algorithm

K.J. Batenburg W.J. Palenstijn

Leiden Institute of Advanced Computer Science (LIACS), Universiteit Leiden
P.O. Box 9512, 2300 RA Leiden, The Netherlands
{kbatenbu, wpalenst}@math.leidenuniv.nl

Abstract

The examination timetabling problem is a difficult combinatorial problem which has to be tackled several times a year by universities all over the world. The multi-stage memetic algorithm introduced in [1] appears to perform very well on various publicly available test datasets. In this paper we analyze an alternative multi-stage algorithm, based on modern AI techniques such as evolutionary computation and tabu search. We show that for large problems using a parallel variant of tabu search leads to significant improvements in the resulting timetable in comparison with the memetic algorithm. Although the new approach causes longer runtimes it can be parallellized very efficiently.

1 Introduction

The exam(ination) timetabling problem is a difficult combinatorial problem which has to be tackled several times a year by universities all over the world. The problem consists of allocating a number of exams in which students participate, to timeslots in such a way that no student has two or more exams in the same timeslot. In addition to this hard constraint some secondary constraints may also be imposed, for example that the number of students having two exams in consecutive timeslots should be minimized. A review of the work published on this problem can be found in [3].

Numerous AI approaches to solving the exam timetabling problem have been proposed, such as tabu search [4, 6] and evolutionary algorithms [1, 2]. On some publicly available datasets [7] the multi-stage memetic algorithm introduced in [1] appears to outperform all other algorithms that it has been compared with. In particular its results on the largest test sets are very favourable. To allow for a good comparison of our new results with the results of the multi-stage memetic algorithm we use a similar problem formulation, described in Section 2.

The multi-stage approach can be applied to other search methods than memetic evolutionary algorithms without major modification. We have designed such a modification, which uses multiple parallel tabu searches for scheduling the events within a single stage. Although using tabu search for generating timetables has been studied before, there has been little research on integrating it into the multi-stage framework.

In Section 3 we describe the algorithm which we have implemented. We compare our algorithm, which is based on tabu search, with the memetic algorithm

from [1]. In Section 4 we show that our approach does indeed lead to significantly improved results for large test cases.

2 Problem definition

We are concerned with solving an instance of the exam timetabling problem. In order to allow for a good comparison between our results and those from [1], we have decided to use the same problem definition as [1].

The input of the problem consists of a set of exams and, for each pair of exams (i, j), the number c_{ij} of participants that they share. Each exam has a length of one time unit. The number of available time units is fixed in advance. The maximal number of people that can work simultaneously (possibly on various exams) is given by a constant number. The problem consists of finding an assignment of exams (*events*) to time units (*timeslots*) in such a way that no person has to participate in two simultaneous events. We do not require each event to be scheduled. However, there is a severe penalty for not doing so. In addition, we want to minimize the number of times a person has two exams in consecutive timeslots, weighted by the time between the two consecutive timeslots (none, a night, a weekend).

Suppose we have a timetable T. Let d_{ij} be the weight for the time between exams i and j in T ($d_{ij} = 3$ if events i and j are on the same day, $d_{ij} = 1$ if they are on consecutive days, $d_{ij} = 0$ otherwise). Then the penalty for T is given by

$$\sum_{\text{all event pairs } (i,j)} c_{ij} d_{ij} \quad + \quad 5000 \times (\text{number of unscheduled events}).$$

The problem is to minimize this penalty function over all timetables that satisfy the hard constraints. For a more formal problem definition, we refer to [1].

3 Implemented algorithms

As a starting point for the implementation of our algorithm we used the multi-stage memetic algorithm from Burke and Newall, described in [1]. We did not make any changes to the multi-stage architecture of the algorithm. It was shown that using this architecture results in significant improvements over the single-stage version.

Our modification consists of using a different search algorithm for scheduling the events in a single stage. In this section we will first give a short description of the original algorithm on which our algorithm is based. Next, we will describe our own search algorithm.

3.1 Original algorithm

The multi-stage memetic algorithm operates on a population of partial timetables (called *individuals*) that are filled in several stages. In each stage a subset of the events is selected and this set is subsequently scheduled in all timetables.

The original algorithm schedules these events using a memetic algorithm. For an extensive description we refer to the original paper. We will only discuss the multi-stage framework here.

In the first stage the first group of events is scheduled, in the next stage the second group, etc. By making this division the number of events that is scheduled in a single stage remains limited, allowing for a thorough search of the solution space in a stage. The choice which events to schedule in each stage is made according to a heuristic. For the experiments we have used the *Saturation Degree* heuristic, which was shown by Burke and Newall to perform very well. The general idea is to schedule the "hard" events — which are heavily constrained — in the first stages and the "easier" events — which are less constrained — in the later stages of the algorithm. By a "heavily constrained event" we mean an event that shares common students with a large number of other events.

The set of events that is selected for scheduling in the current stage is called the *active set*. The active set changes in every stage. After events have been removed from the active set, their timeslot has been fixed permanently. An obvious drawback of the multi-stage approach is that in each stage only a limited set of events is considered. To partially compensate for this loss of flexibility, half the events in the active set will remain in the active set of the next stage. This subset is called the *look-ahead*. These events are scheduled along with the events of the current stage, but the timeslots assigned to look-ahead events may still change in the next stage, after which they will certainly be fixed. The other half of the active set is removed after the current stage. Therefore these events have been assigned their final timeslot. Note that different individuals may correspond to different assignments of events to timeslots. Evolution of the population of timetables determines how these partial timetables from previous stages spread through the population. For scheduling the events in the active set various algorithms can be used. In [1] a memetic algorithm is used. We use a parallel tabu search algorithm instead.

3.2 Search algorithms

In order to analyze the results of the original memetic algorithm, we reimplemented the algorithm. The search algorithm that is used for scheduling the events in a single stage is a memetic evolutionary algorithm with two mutation operators. Cross-over is not used.

We obtained results that were quite similar to the results reported in the original paper. Our results will be detailed in the next section, where we present the experiments. We noticed that the memetic algorithm converges very fast, not exploring a large part of the search space. In order to improve the explorative power of the search algorithm we experimented with different search algorithms, based on tabu search.

3.2.1 Tabu search

We use tabu search as our main search algorithm. For an overview of tabu search we refer to [5].

The neigbourhood of the current timetable depends on a positive integer parameter d_{max}. It consists of all timetables that can be reached by first unscheduling $d \leq d_{max}$ events and subsequently rescheduling these events. For each event, all timeslots are considered in random order, and the first allowed timeslot is chosen. By using this neighbourhood definition the tabu search can take both medium-sized steps (rescheduling several events) and small steps (rescheduling only one or two events).

In order to prevent the algorithm from being trapped in a local optimum, tabu search uses a tabu list of forbidden moves. In our context, we define a move as a change in the assigned timeslot of a single event to a certain other timeslot. Getting from the current timetable to one of its neighbours can involve a number of such moves. The sequence of event-moves that changes the current timetable into the selected neighbour consists of a number of event-moves which unschedule the events involved, followed by the same number of event-moves which reschedule the events involved. All these event-moves, including the unscheduling moves, become tabu. Such tabu moves may not be used for moving from the new current timetable to one of its neighbours unless the neighbour represents the best timetable found so far. A reverse event-move keeps its tabu status until a certain number of new event-moves has been added to the tabu list. This number of event-moves is chosen randomly from the interval $[r_{min}, r_{max}]$. Using this approach results in a variable length list of tabu moves.

Because the neighbourhood that we defined can consist of a huge number of timetables, in each iteration a random fixed-sized subset of the neighbourhood is considered. The best non-tabu neighbour from this set is chosen as the new timetable.

If a certain number of iterations has passed without improving upon the best solution found so far, the tabu list is cleared. Directly after the clearing of the tabu list the local search in the current area is intensified, because all event-moves are allowed. After a number of iterations the search diversifies, because the tabu list is gradually refilled.

3.2.2 Parallel tabu search

The main drawback of the memetic algorithm used in [1] is the early convergence of the population to a single solution. In combination with the inflexibility that results from using the multi-stage framework, this severely limits the part of the search space that is explored.

A good search algorithm should not only explore a significant part of the search space in each stage, but it should also allow for some flexibility in the scheduling of the events in earlier stages. This flexibility is incorporated in the evolutionary algorithm by using a population of timetables instead of a single timetable. We can apply the same principle to tabu search: maintain a population of timetables and apply a tabu search to each of them. Because all tabu searches are independent we may expect the set of timetables to diverge, instead of converge, even when starting with identical timetables. At the end of each stage a selection criterion is applied, specifying which partial timetables progress to the next stage. The best

half of the population is duplicated and the other half is discarded. Because of this we have chosen the population size to be even. The operation of the algorithm is shown in Figure 1.

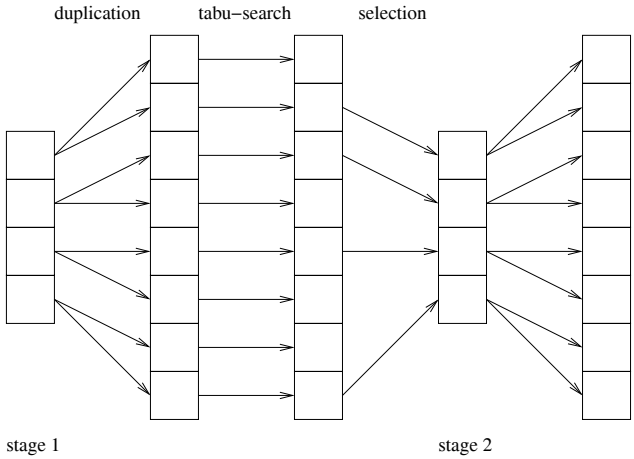


Figure 1: Operation of parallel tabu search

Suppose the population size is 2^p where p is an integer greater than 0. Then the best partial timetable resulting from stage s will need at least $\lg(2^p) = p$ stages before it can dominate the population, as shown in Figure 2. Therefore the algorithm maintains population variety among at least the events scheduled in the last p stages.

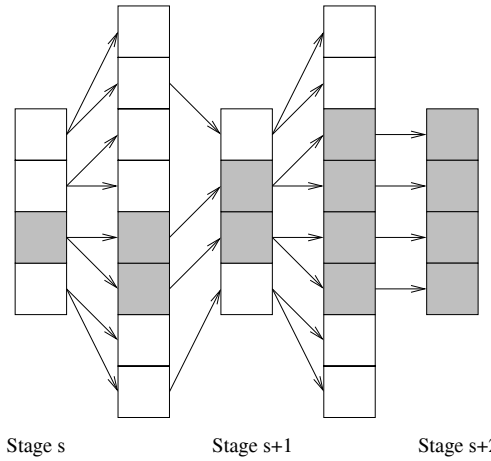


Figure 2: Propagation of a partial timetable through the population

Of course the runtime is a major concern when applying this strategy. Therefore the population size must be kept fairly low. Because the tabu searches in a

single stage can be performed in parallel we can dramatically decrease the runtime by using parallel processors for the computation. However, we have only implemented a sequential version. It requires considerably more time than the memetic algorithm. After performing some experiments we observed that the quality of the final solution is largely determined by the scheduling of events in the first few stages. In order to lower the runtime for the largest dataset we strongly decrease the population size after a small number of stages. We will describe this in more detail in the next section.

4 Experimental results

For testing the algorithms we used three publicly available test data sets from the Timetabling data repository of the University of Toronto [7]: *car-f-92*, *kfu-s-93* and *pur-s-93*. These data sets provide lists of exams and their participating students, but leave other parameters, such as the number of available timeslots and the maximal number of people taking exams simultaneously, unspecified. We use the same values for these parameters as used in [1]. Table 1 shows some characteristics of the three test data sets that we have used. It shows the number of events, the number of timeslots, the number of students, the maximal number of people that can work simultaneously and the density of the conflict matrix. This density is defined as the number of event-pairs which share students, divided by the total number of event-pairs.

We have performed all experiments on an AMD Athlon XP 1700, running Linux. The algorithms have been implemented in C++, using the GNU gcc compiler.

We implemented the memetic algorithm from Burke and Newall in order to analyze its performance. Although the details of our implementation will probably be different from the original implementation, the quality of the resulting timetables is similar to the results reported by Burke and Newall. Our experiments with this algorithm show that the population converges quickly to a state in which all individuals are nearly equal. Obviously from that point on there will hardly be any improvement of the best solution found. Increasing the population size (resulting in increased runtimes) did not improve the solution quality significantly. Our algorithm is an attempt to diversify the search process.

instance	#events	#timeslots	#students	max. simult.	confl. density
car-f-92	543	36	55,552	2,000	0.14
kfu-s-93	461	21	25,118	1,955	0.06
pur-s-93	2,419	30	120,690	5,000	0.03

Table 1: Some characteristics of the test data sets

We will now describe the parameters that we used for each of the algorithms. The original algorithm from Burke and Newall was shown to perform best with active set sizes between 50 and 100, using a look-ahead set. In Table 2 their results are shown, obtained with the best reported parameter settings for each of the test

cases. In order to allow for a comparison of the runtime with our own algorithms the table shows the runtime of our implementation of the original algorithm.

For the parallel tabu search algorithm we have experimented with various parameter settings. In order to determine the final settings we have performed several preliminary experiments. We will now describe the final settings. We used an active set size of 130. The tabu search is stopped when no improvement on the best result has been found in the last 500 iterations. When an event-move is inserted into the tabu list, the duration of the tabu period for this event-move is determined. A random number r between 500 and 1500 is generated. The event-move remains tabu until r new event-moves have been added to the tabu list. In each iteration a random subset of the neighbourhood of the current timetable is generated. The total number of generated neighbours is limited to 10000. The neighbours are selected by first generating a random positive integer $d \leq d_{max} = 3$ and then rescheduling d events, as described in Section 3.2.1. The tabu list is cleared if no improvement on the best result has been found after 75 iterations. We observed that the performance of the tabu search algorithm is almost completely determined by the performance in the first few stages. We have used this property to dramatically decrease the runtime of parallel tabu search by strongly decreasing the "population size" after the first four stages. After the fourth stage the population size is decreased to just 2 timetables. For the *car-f-92* and *kfu-s-93* datasets we used a population size of 16 timetables for the first four stages. Because the *pur-s-93* dataset is significantly more demanding, we decided to use a larger population size, 24, for this dataset.

instance	algorithm	avg. time (s)	avg. penalty	best	
				unsched.	consec. pen.
car-f-92	Burke and Newall	59	1,765	0	1,665
	Parallel tabu	6,312	1,649	0	1,609
kfu-s-93	Burke and Newall	38	1,608	0	1,588
	Parallel tabu	5,964	1,460	0	1,351
pur-s-93	Burke and Newall	419	65,461	9	12,246
	Parallel tabu	13,668	17,065	3	10,889

Table 2: Experimental results

We have run both algorithms five times on each test case. Table 2 lists the average runtime and the average penalty obtained by the algorithms. It also shows the penalty of the best of the five solutions, decomposed into the number of unscheduled exams and the penalty caused by students having consecutive exams. Increasing the population size for the algorithm of Burke and Newall did not result in improved solution quality, even when the runtime was equal to the runtime of the parallel tabu search. Because shorter runtimes are preferable, we omitted the long runs from the table.

5 Conclusions

Although the memetic algorithm from [1] works very fast on all test instances there is still much room for improvement on large test cases such as *pur-s-93*. It converges rapidly in every stage, limiting the part of the solution space that is thoroughly searched. Unfortunately, the *pur-s-93* dataset is the only freely available set of its size. Therefore we had to focus on this set in our test results.

We have shown that using a parallel tabu search instead of a memetic algorithm leads to significantly improved timetables for this test case, at the expense of extra runtime. The original algorithm does not perform better when run with a larger population, resulting in a longer runtime. Performing a parallel tabu search takes a long time when using a single processor, but it can be parallellelized efficiently. The number of unscheduled events was reduced to one-third of the unscheduled events in the best solution found by the memetic algorithm.

6 Acknowledgments

The authors would like to thank Walter Kosters for making numerous valuable suggestions.

References

[1] E.K. Burke and J.P. Newall. A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, Vol. 3.1:63–74, 1999.

[2] E.K. Burke, J.P. Newall, and R.F. Weare. A memetic algorithm for university exam timetabling. In *PATAT 95 — Proc. of the First Int. Conf. on the Practice and Theory of Automated Timetabling*, pages 241–250, 1995.

[3] E.K. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research (EJOR)*, Vol. 140:266–280, 2002.

[4] L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In *PATAT 2000 — Proc. of the Third Int. Conf. on the Practice and Theory of Automated Timetabling*, pages 104–117, 2000.

[5] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[6] G.M. White and B.S. Xie. Examination timetables and tabu search with longer term memory. In *PATAT 2000 — Proc. of the Third Int. Conf. on the Practice and Theory of Automated Timetabling*, pages 85–103, 2000.

[7] Online timetabling data repository of the University of Toronto, `ftp://ie.utoronto.ca/mwc/testprob`.

Context-Enhanced Object Detection in Natural Images

N.H. Bergboer E.O. Postma H.J. van den Herik

Department of Computer Science
University of Maastricht
P.O. Box 616
6200 MD Maastricht, The Netherlands
{n.bergboer,postma,herik}@cs.unimaas.nl

Abstract

We present a novel machine-learning method to detect the locations and sizes of objects in still images. Unlike many existing methods, our context-enhanced method takes the spatial context of objects into account. The detection proceeds in two phases. In the first phase, contexts that are likely to contain instances of a predefined object class are selected from a color differential structure. In the second phase an object-detection method based on overcomplete wavelets searches for objects within the selected contexts only. We applied the method to the detection of human faces. The results show that the use of context reduces the number of false positives by a factor of 7.3 and reduces the search space by a factor of 15.7. From the results, we may conclude that context-enhanced object detection is a viable means to reduce the number of false positives and the size of the search space in still images.

1 Introduction

This paper presents a novel context-enhanced object-detection method. Object detection is the automatic determination of the locations and sizes of objects in the framework of an image, where the objects belong to a predefined class. Object detection is an active research area that has yielded many methods (e.g., [8, 10, 12]). Unfortunately, these methods suffer from two main disadvantages. First, they use brute-force search strategies that impose large computational costs. Second, they result in relatively high false-detection rates. Recent work by Torralba and Sinha [15] shows that the use of an object's spatial context can reduce the search space. However, Torralba and Sinha do not combine their context-selection method with an object-detection stage. In this paper we propose an integrated method for detecting objects within contexts in which we do combine context and object detection. Like Torralba and Sinha, we show that the search space is reduced considerably. But more importantly, we further show that the use of context reduces the number of false detections.

The paper is organized as follows. Section 2 introduces our context-enhanced method. Section 3 applies the method to the detection of frontal human faces, and supplies implementation details. In Section 4 we present results, which are

discussed in Section 5. In Section 6 we conclude upon the efficiency of the context-enhanced object-detection method.

2 Description of the context-enhanced method

We start by describing two observations that led to our method. Then, the method is discussed in more detail. The first observation comes from human perception studies. There, it is a well-established fact that context is important to observers when locating and detecting objects [1]. This importance can be readily illustrated. Figure 1 shows two examples; the small square images are enlarged versions of the square regions indicated by boxes in the large images. Considered in isolation, both small images are highly similar to faces. When considered in their natural context, the interpretation as faces is immediately suppressed.

Figure 1: Examples of patterns that are similar to faces, but that are clearly not faces when viewed in their context.

The second observation comes from computer vision. There, most current object-detection methods do not use context, and implicitly assume that the features related to an object's context are not related to the features of the object itself [5]. It is remarkable that the spatial context is generally ignored in computer vision. For instance, in video data the temporal context is generally acknowledged to be important [4], but the spatial context is neglected. In still images, recent research by Torralba and Sinha shows that the features of an object's context are statistically correlated with those of the object itself [15]. However, they claim that the use of context serves no other purpose than the reduction of the search space for objects. As against this, we expect the use of context to be beneficial to the object-detection performance both in terms of search space reduction and in terms the number of falsely detected objects.

Our context-enhanced object-detection method proceeds in two stages. The first stage is a context-detection stage in which an image is scanned at a coarse scale for likely spatial contexts of the object class of interest. This stage is illustrated in the left panel of Figure 2. It results in the selection of a number of context regions. The second stage is an object-detection stage in which objects are searched for and detected at a finer scale within the center parts of selected context regions only. This stage is illustrated in the center panel of Figure 2. The right panel of the Figure is the final state of the second stage; it shows the final detection result.

Many existing object-detection methods use window sliding techniques [8, 10, 12]. In these techniques, a square window of a given size is slid over the image. To capture all relevant objects, sliding is performed for different scales. At each position and scale, the contents of the window are classified as object or non-object.

In our method, a window-sliding technique is initially used for the context classifier. To this end, we define an object's spatial context as a square region

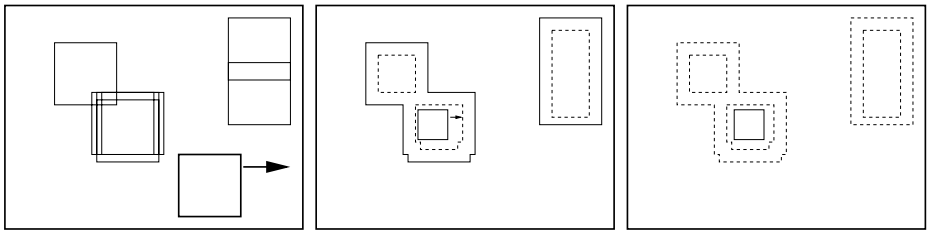


Figure 2: Graphical representation of the context-enhanced detection method at a given scale.

of the image, of which the lateral dimensions are twice that of the square region enclosing the object. Subsequently, a sliding window is used for the object classifier in the context regions only. In both detection stages, classification is based on visual features that are meaningful in terms of human vision [7]. The choice of features is essential for obtaining a good classifier. Appropriate visual features for both contexts and objects are discussed in the next subsections.

2.1 Visual features for the context-detection stage

One of the earliest stages in human vision is the detection of edges in the retinal image. In addition, the detection of color is generally acknowledged to be important for the analysis of images [7]. We therefore use color edges as visual features.

Color edges can be conveniently extracted from a color differential structure by using Geusebroek *et al.*'s [3] scale-space theory. These authors decompose an image into three color receptive fields; an intensity field $\mathcal{R}_1$, a yellow-blue opponent field $\mathcal{R}_2$ and a red-green opponent field $\mathcal{R}_3$. This decomposition is biologically inspired since the decomposition of color data into opponent fields occurs in the early stages of the human visual system [7]. The transformation from RGB values to the color receptive fields can be conveniently written as a linear transformation adopted from [13]:

$$\begin{bmatrix} \mathcal{R}_1 \\ \mathcal{R}_2 \\ \mathcal{R}_3 \end{bmatrix} = \begin{bmatrix} 0.002358 & 0.025174 & 0.010821 \\ 0.011943 & 0.001715 & -0.013994 \\ 0.013743 & -0.023965 & 0.00657 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

The color receptive fields are subsequently used to calculate the color differential structure $\mathcal{G}$ for yellow-blue transitions and $\mathcal{W}$ for red-green transitions as follows [13]:

$$\mathcal{G} = \frac{1}{\mathcal{R}_1^2} \sqrt{\left(\mathcal{R}_2 \frac{\partial \mathcal{R}_1}{\partial x} - \mathcal{R}_1 \frac{\partial \mathcal{R}_2}{\partial x} \right)^2 + \left(\mathcal{R}_2 \frac{\partial \mathcal{R}_1}{\partial y} - \mathcal{R}_1 \frac{\partial \mathcal{R}_2}{\partial y} \right)^2} \tag{2}$$

$$\begin{aligned} \mathcal{W} = \frac{1}{|\mathcal{R}_1^3|} \Bigg[& \left(2\mathcal{R}_2^2 \frac{\partial \mathcal{R}_1}{\partial x} - 2\mathcal{R}_1 \mathcal{R}_2 \frac{\partial \mathcal{R}_2}{\partial x} + \mathcal{R}_1 \left(-\mathcal{R}_3 \frac{\partial \mathcal{R}_1}{\partial x} + \mathcal{R}_1 \frac{\partial \mathcal{R}_3}{\partial x} \right) \right)^2 \\ & + \left(2\mathcal{R}_2^2 \frac{\partial \mathcal{R}_1}{\partial y} - 2\mathcal{R}_1 \mathcal{R}_2 \frac{\partial \mathcal{R}_2}{\partial y} + \mathcal{R}_1 \left(-\mathcal{R}_3 \frac{\partial \mathcal{R}_1}{\partial y} + \mathcal{R}_1 \frac{\partial \mathcal{R}_3}{\partial y} \right) \right)^2 \Bigg]^{1/2} \end{aligned} \tag{3}$$

In these expressions, all partial derivatives with respect to x and y are Gaussian derivatives at scale σ [13]. The value of the scale parameter σ is set in accordance with the coarse scale at which the context selection takes place.

In our method we omit the normalization, i.e. the $1/\mathcal{R}_1^2$ and $1/|\mathcal{R}_1^3|$ terms in (2) and (3), respectively. The reason for omitting is to prevent the amplification of artifacts, in regions of low intensity. These artifacts are present in images for two reasons. First, pixel-values in digital images are discretized. Second, many images are generated by lossy compression techniques, such as JPEG. We denote the resulting non-normalized quantities as $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{W}}$.

2.2 Visual features for the object-detection stage

To detect objects we adopt the method proposed by Papageorgiou and Poggio [8], who employ features derived from an overcomplete wavelet basis at two scales to detect various object classes, i.e., human faces, pedestrians, and automobiles. Papageorgiou and Poggio use grayscale and pseudo-color wavelets. The choice for grayscale or pseudo-color depends on the object class of interest.

3 Application to human face detection

In order to assess the performance of our context-enhanced object-detection method we apply it to the detection of a single class of objects: frontal human faces. The class of faces is an appropriate challenge for our method because it is well documented and data for comparison is readily available [5]. In order to obtain training data, a total of $3,383$ faces were manually labeled in 995 color images obtained from the Internet. The coordinates of the facial context regions are derived by scaling the face regions by a factor of 2. An additional set of $1,900$ positive instances representing frontal faces is obtained by labeling unoccluded faces in the AR faces database [6].

3.1 Detection of facial contexts

For frontal human faces, we use a context region of which the size of roughly twice the size of a face; a sliding window of size 40×40 pixels is used. In order to extract features for detection, we choose to set the scale of the Gaussian derivatives to $\sigma = 8$. Larger scales provide very coarse information only which is not helpful in distinguishing between contexts and non-contexts. Conversely, smaller scales capture mostly intra-class variance rather than inter-class variance.

A context of size 40×40 pixels at one scale generates $1,600$ features for both $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{W}}$. For practical reasons, we down-sample $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{W}}$ to the 10×10 matrices $\widetilde{\mathcal{G}}_D$ and $\widetilde{\mathcal{W}}_D$, and normalize $\widetilde{\mathcal{G}}_D$ and $\widetilde{\mathcal{W}}_D$ by their mean values $\overline{\mathcal{G}}_D$ and $\overline{\mathcal{W}}_D$, respectively. We effectively assume the area of interest to be uniformly illuminated. The feature vector $\theta_C \in \mathbb{R}^{200}$ is constructed by vectorizing and concatenating the normalized components.

The color differential structure components $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{W}}$ are calculated for the set of labeled positive contexts, and the average is displayed in Figure 3. Yellow-blue transitions $\widetilde{\mathcal{G}}$ are displayed on the left, together with the down-sampled version $\widetilde{\mathcal{G}}_D$. Red-green transitions $\widetilde{\mathcal{W}}$ and the downsampled $\widetilde{\mathcal{W}}_D$ are shown on the right. It is clear that while the yellow-blue transitions $\widetilde{\mathcal{G}}$ mainly take place in the area

between the chin and the neck, the red-green transitions $\widetilde{\mathcal{W}}$ encode information in the area of the cheeks and forehead.

Figure 3: Color differential structure features for the facial context averaged over the set of positive training contexts.

3.2 Detection of faces

For the detection of faces we adopt the method of Papageorgiou and Poggio with grayscale wavelets, as these yield the most compact representation of the object class [8]. A sliding window of size 19×19 pixels is used and the feature set consists of wavelet coefficients for square-support Haar wavelets of size 2×2 and 4×4. We use an overcomplete representation of 17×17 coefficients for a given scale and orientation. For both scales, the low-pass filtered data is discarded and the absolute value of only the horizontal, vertical and diagonal detail coefficients are retained. Each of the resulting six components is normalized by its mean and a feature vector $\theta_O \in \mathbb{R}^{1,734}$ is obtained by vectorizing and concatenating the six components.

3.3 Training

A support vector machine (SVM) is used for classification of both contexts and objects, as SVMs perform well and can be efficiently trained on large datasets [11]. In addition, efficient software is readily available [2, 9]. A quadratic kernel is used, as results by Papageorgiou and Poggio [8] indicate that such a kernel yields good object-detection performance.

For the context classifier, we use a set of $2,000$ positive context instances from the $3,383$ labeled contexts in the images obtained from the internet. A set of $20,000$ negative instances is obtained by extracting features from randomly-selected regions in these same images. The classifier trained on these data uses $3,764$ support vectors.

For the object classifier we use a modification of the "boot-strap training" strategy proposed by Sung and Poggio [12]. First, an initial training set is constructed that consists of $1,900$ positive instances from the AR faces database and $19,000$ negative instances from random regions in the set of images obtained from the Internet. A classifier is then trained and run on 60 images, which yields $31,613$ false positives that are added to the set of negative training instances. Subsequently, a classifier is trained on this set, and it uses $2,036$ support vectors.

4 Results

The performances of the trained classifiers are assessed on 265 images from the Internet that together contain 501 labeled faces. The images contain labeled faces that are at least 60×60 pixels in size. To ensure that all faces are found, the images

are classified for face sizes of 45×45 pixels and up, and for context sizes of 90×90 pixels and up. The scale is varied logarithmically such that subsequent scales differ by a factor of 1.1. To assess the benefits of using our context-enhanced method, the images are also scanned for faces by a brute-force method which detects faces at all image locations.

In order to assess the detection quality, we distinguish three cases. Labelled faces can be detected (*true positive*), missed (*false negative*), or falsely detected (*false positives*). From the object-windows that are classified positively, at least one should overlap sufficiently with a labeled face in order for that face to be a *true positive*; the labeled face and the detected window should cover at least halve of each other's area. Otherwise, the labeled face is a *false negative*. Detected windows that do not overlap sufficiently with any of the labeled faces are regarded as *false positives*.

The results for the brute-force method and our context-enhanced method are displayed in Table 1 in terms of detection rates and false positive rates. In addition, the average number of false positives per image is listed for image dimensions 800×600. Figure 4 shows two graphs obtained by varying the output threshold of the context classifier. The graph on the left is a ROC curve which shows the trade off between detection rate and false positive rate. The graph on the right shows the trade off between detection rate and the ratio between the search space after context selection and the brute force search space. In both curves, the output threshold on the context classifier is varied from $-\infty$ (no selection, equivalent to the brute-force method) to $+3$ (strict selection). The values in Table 1 correspond to a threshold of 0. The graphs show that a stricter context selection causes a smaller search space and false-positive rate, at the cost of a smaller detection rate.

		False positive	Average per 800×600 image	
Method	*Detection rate*	*rate*	*False positives*	*Search space*
Brute-force	449/501 (89.6%)	$3.0 \cdot 10^{-5}$	11	$3.79 \cdot 10^5$
Context-enhanced	355/501 (70.8%)	$4.1 \cdot 10^{-6}$	1.6	$2.42 \cdot 10^4$

Table 1: Detection results for the brute-force face detection method of Papageorgiou and Poggio [8] (brute-force) and our context-enhanced method.

Our context-enhanced method yields a decrease in false positives by a factor 7.3 as compared to the brute-force method. In addition, the search space of the object classifier is reduced by a factor of 15.7. The lowered detection rate (from 89.6% to 70.8%) is not surprising. The use of context restricts the system's detection capabilities to likely contexts, thus necessarily missing those objects embedded in unlikely contexts. Stated differently: the detection rate for objects is bounded by the detection rate of contexts. The detection rate for contexts is necessarily lower than that of objects for two reasons. First, the intra-class variance for contexts is higher than that for objects because variations in the object's surroundings are larger than variations within the object itself. Second, we use a low-dimensional context representation that, while being fast, worsens the separability between positive and negative context instances.

Figure 5 illustrates typical detection results of our context-enhanced method. The top row of images shows the contexts detected by the context classifier, and

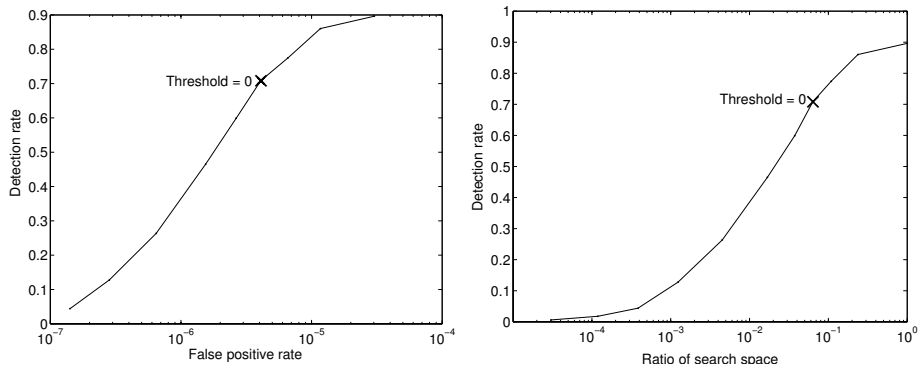


Figure 4: ROC curve (*left*) and ratio of the search space to be searched for objects after selecting contexts (*right*).

the bottom row shows faces detected by our method (*solid*) and faces that have additionally been found in the brute force method (*dashed*). From left to right, Figure 5 shows an example in which our method works well, an example of a false negative due to the use of context, and an example of a false positive despite the use of context.

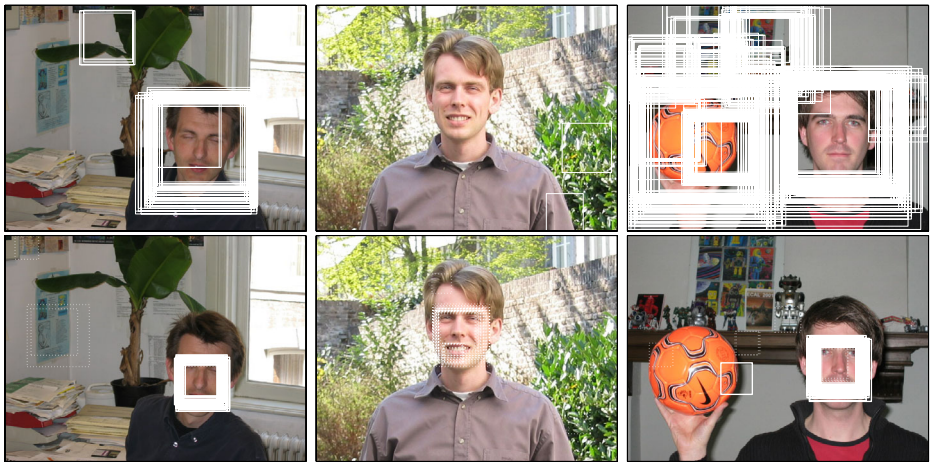


Figure 5: Example results for our method.

5 Discussion

Our context-enhanced method reduces the number of false positives at the cost of a small decrease in detection rate. While this trade-off is common to many detection methods [5, 8, 10, 12], our method achieves the same result with a considerable reduction of the search space. The associated reduction of search time is a strategy commonly employed by humans [1]. Our method is best used in situations where quick results are needed and a few false negatives can be tolerated.

Our method can be further improved by (1) exploiting other contextual cues, such as the location of the horizon or the average image depth [14] and (2) by

applying a "soft" selection (i.e. probablistic selection) of the relevant context. These will be topics of future research.

6 Conclusions

We proposed an object detection method that uses context to reduce the number of false object detections and to reduce the search space. Human face-detection experiments have shown that using context reduces the false positive rate and the search space by a considerable factor. Hence we tentatively conclude that context enhanced object detection is an effective method for reliable and fast detection.

Acknowledgements

This research is carried out within the Netherlands Organization for Scientific Research (NWO) Token 2000 project Eidetic (grant 634.000.001).

References

[1] I. Biederman. On the semantics of a glance at a scene. In M. Kubovy and J. Pomerantz, editors, *Perceptual organization*. Erlbaum, Hillsdale, NJ, 1981.

[2] G. C. Cawley. MATLAB support vector machine toolbox (v0.50β) (`http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox`). University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ, 2000.

[3] J. M. Geusebroek, A. Dev, R. van den Boomgaard, A. W. M. Smeulders, F. Cornelissen, and H. Geerts. Color invariant edge detection. In *Scale-Space theories in Computer Vision*, volume 1252 of *Lecture Notes in Computer Science*, pages 459–464. Springer-Verlag, 1999.

[4] S. Gong, S. J. McKenna, and A. Psarrou. *Dynamic Vision, From Images to Face Recognition*. Imperial College Press, London, 2000.

[5] E. Hjelmås and B. K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274, September 2001.

[6] A. M. Martinez and R. Benavente. The AR face database. Technical Report CVC #24, Electrical and Computer Engineering, Purdue University, June 1998.

[7] S. E. Palmer. *Vision Science, Photons to Phenomenology*. MIT Press, Cambridge, Massachusetts, 1999.

[8] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.

[9] R. Rifkin, M. Nadermann, and P. Moreno. Svmfu: A fast, flexible support vector machine classification algorithm. Research abstract, MIT CBCL, 2001.

[10] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2000.

[11] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 1st edition, December 2001.

[12] K. K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[13] B. M. ter Haar Romeny. *Front-End Vision and Multi-Scale Image Analysis*. Kluwer Academic Publishers, 2002.

[14] A. Torralba and A. Oliva. Depth estimation from image structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1226–1238, September 2002.

[15] A. Torralba and P. Sinha. Statistical context priming for object detection. In *Proceedings of the International Conference on Computer Vision*, Vancouver, Canada, 2001.

Game Specification in the Trias Politica

Guido Boella[a] Leendert van der Torre[b]

[a] Dipartimento di Informatica - Università di Torino - Italy
[b] CWI - Amsterdam - The Netherlands

Abstract

In this paper we formalize the specification of games in the trias politica using Rao and Georgeff's specification language BDI_{CTL^*}. In particular, we generalize Rao and Georgeff's specification of single agent decision trees to multiagent games, for which we introduce observations and recursive modelling, in this setting we formalize obligations, and we characterize four kinds of agents, called legislators, judges, policemen and citizens. Legislators are characterized by their power to create and revise obligations, judges are characterized by their power to count behavior of citizens as violations, and policemen are characterized by their ability to sanction behavior.

1 Introduction

Montesquieu's trias politica distinguishes between three autonomous powers. Due to their autonomy, these powers can be analyzed as agents and mental attitudes can be attributed to them. In this paper we call these agents legislators, judges and policemen. Moreover, our multiagent model contains a set of agents called citizens. Various games with obligations can be played in this multiagent system. E.g., possible decision problems are:

- A citizen considers whether it will fulfill or violate the obligations, given its expectations of judges and policemen [2, 3].
- A legislator considers which sanctions it associates with norms, such that citizens will fulfill obligations [4].

We are interested in the specification of such decision problems in the trias politica. To take the limited rationality of the agents into account we use recursive modelling [6] instead of equilibria analysis, the most popular approach in game theory. We use Rao and Georgeff's BDI_{CTL^*} and generalize their translation to BDI_{CTL^*} models of single agent decision trees [8] to multiagent games. In this approach, belief accessible worlds represent possible alternatives, desire (or goal) accessible worlds represent the alternatives the agent takes into consideration, and intention accessible worlds represent the optimal decisions. Our research question breaks down into the following questions.

1. How to formally specify recursive games in BDI_{CTL^*}?
2. How to formally specify obligations in BDI_{CTL^*}?
3. How to characterize legislators, judges and policemen in BDI_{CTL^*}?

The motivation of our research questions is the specification of decision problems of the kind introduced by Boella and Lesmo [2].Normative systems that control and regulate

behavior like legal, moral or security systems are autonomous, they react to changes in their environment, and they are pro-active. For example, the process of deciding whether behavior counts as a violation is an autonomous activity. Since these properties have been identified as the properties of autonomous or intelligent agents [12], normative systems may be called *normative agents*. This goes beyond the observation that a normative system may contain agents, like a legal system contains legislators, judges and policemen, because *a normative system itself is called an agent*. The first advantage of the normative systems as agents perspective is that the interaction between an agent and the normative system which creates and controls its obligations can be modelled as a game between two agents. The second advantage of the normative systems as agents perspective is that, since mental attitudes can be attributed to agents, we can attribute mental attitudes to normative systems. The logic proposed in this paper can be used to formally specify games defined in [2, 3, 4] at a high level of abstraction.

Rao and Georgeff do not present a full axiomatization of the formalization of decision trees in their logic. For our much more complicated system we focus also on semantics and we do not present a full axiomatization. However, we do show some interesting properties which characterize important mechanisms of the logic. For example, the characteristic property of recursive modelling is that for each agent the decision of later agents is fixed.

The layout of this paper is as follows. In Section 2 we discuss Rao and Georgeff's $\text{BDI}_{\text{CTL}^*}$. In Section 3 we define recursive decision problems in the logic. In Section 4 we define obligations and we characterize legislators, judges and policemen.

2 The logic

In this section we use an equivalent reformulation of Rao and Georgeff's formalism [9] presented by Schild [10]. We only consider the semantics. Following Rao and Georgeff we do not use desires but goals, because that seems to better fit the interpretation of games.

Definition 1 *Assume n agents. The admissible formulae of BDI$_{CTL^*}$ are categorized into two classes, state formulae and path formulae.*

- *S1 Each primitive proposition is a state formula.*
- *S2 If α and β are state formulae, then so are $\alpha \wedge \beta$, $\neg\alpha$.*
- *S3 If α is a path formula, then $E\alpha$, $A\alpha$ are state formulae.*
- *S4 If α is a state formula and $1 \leq i \leq n$, then $B_i(\alpha), G_i(\alpha), I_i(\alpha)$ are state formulae as well.*
- *P1 Each state formula is also a path formula.*
- *P2 If α and β are path formulae, then so are $\alpha \wedge \beta$, $\neg\alpha$.*
- *P3 If α and β are path formulae, then so are $X\alpha$, $\alpha U \beta$.*

The unary operator $\Diamond$ (eventually) is defined as a special case of the binary U operator by $\Diamond\alpha = \top U \alpha$, while $\Box$ (always) is the dual of $\Diamond$ by $\Box\alpha = \neg\Diamond\neg\alpha$. Finally, $\wedge$ and $\rightarrow$ are defined as usual.

The semantics of $\text{BDI}_{\text{CTL}^*}$ involves a modal and a temporal dimension. The truth of a formula depends on both the possible world w and the temporal state s. A pair $\langle w, s \rangle$

is called a situation in which $\text{BDI}_{\text{CTL}^*}$ formulae are evaluated. The relation between situations is traditionally called an accessibility relation (for beliefs) or a successor relation (for time).

Definition 2 *Assume n agents. A Kripke structure $M = \langle \Delta, \mathcal{R}, \mathcal{B}_1, \mathcal{G}_1, \mathcal{I}_1, \ldots, \mathcal{I}_n, L \rangle$ forms a situation structure if Δ is a set of situations, $\mathcal{R} \subseteq \Delta \times \Delta$ is a binary relation such that $w = w'$ whenever $\langle w, s \rangle \mathcal{R} \langle w', s' \rangle$, $Z_i \subseteq \Delta \times \Delta$ for $Z \in \{B, G, I\}$ and $1 \leq i \leq n$ are binary relations such that $s = s'$ whenever $\langle w, s \rangle Z_i \langle w', s' \rangle$, and L an interpretation function that assigns a particular set of situations to each primitive proposition. $L(p)$ contains all those situations in which p holds.*

A speciality of CTL* is that some formulae – called path formulae– are not interpreted relative to a particular situation. What is relevant here are full paths. The reference to M is omitted whenever it is understood.

Definition 3 *Assume n agents. A full path in situation structure M is an infinite sequence $\chi = \delta_0, \delta_1, \delta_2, \ldots$ such that for every $i \geq 0$, δ_i is an element of Δ and $\delta_i \mathcal{R} \delta_{i+1}$. We say that a full path starts at δ iff $\delta_0 = \delta$. We use the following convention. If $\chi = \delta_0, \delta_1, \delta_2, \ldots$ is a full path in M, then χ^i $(i \geq 0)$ denotes exactly the same infinite sequence as χ, except that the first i components are omitted.*

Let M be a situation structure, δ a situation, and χ a full path. The semantic relation $\models$ for BDI_{CTL^*} is then defined as follows:

S1 $\delta \models p$ *iff* $\delta \in L(p)$.
S2 $\delta \models \alpha \wedge \beta$ *iff* $\delta \models \alpha$ *and* $\delta \models \beta$.
$\delta \models \neg\alpha$ *iff* $\delta \models \alpha$ *does not hold.*
S3 $\delta \models E\alpha$ *iff there is a full path* χ *in* M *starting at* δ *such that* $\chi \models \alpha$.
$\delta \models A\alpha$ *iff for every full path* χ *in* M *starting at* $\delta, \chi \models \alpha$.
S4 $\delta \models B_i(\alpha)$ *iff for every* $\delta' \in \Delta$ *such that* $\delta \mathcal{B} \delta', \delta' \models \alpha$.
$\delta \models G_i(\alpha)$ *iff for every* $\delta' \in \Delta$ *such that* $\delta \mathcal{G} \delta', \delta' \models \alpha$.
$\delta \models I_i(\alpha)$ *iff for every* $\delta' \in \Delta$ *such that* $\delta \mathcal{I} \delta', \delta' \models \alpha$.
P1 *If* α *is a state formula and* χ *starts at* δ, *then* $\chi \models \alpha$ *iff* $\delta \models \alpha$.
P2 $\chi \models \alpha \wedge \beta$ *iff* $\chi \models \alpha$ *and* $\chi \models \beta$.
$\chi \models \neg\alpha$ *iff* $\chi \models \alpha$ *does not hold.*
P3 $\chi \models X\alpha$ *iff* $\chi^1 \models \alpha$.
$\chi \models \alpha U \beta$ *iff there is* $i \geq 0$ *with* $\chi^i \models \beta$ *and for all* j, $(0 \leq j < i), \chi^j \models \alpha$.

$\text{BDI}_{\text{CTL}^*}$ is very general. In particular, we have epistemic states over temporal sequences (e.g., $B_i A \Box p$, i believes that always p) which is called internal dynamics, and temporal sequences over epistemic states (e.g., $A \Box B_i p$, Always $B_i p$ will be the case) which is called external dynamics. Rao and Georgeff discuss realism properties expressing the fact that for each believed world there must be a goal world which is a subtree of it (and analogously for goals and intentions):

$$B_i p \rightarrow G_i p \qquad G_i p \rightarrow \neg B_i \neg p \qquad G_i E \Diamond p \rightarrow B_i E \Diamond p \qquad I_i E \Diamond p \rightarrow G_i E \Diamond p$$

and commitment strategies (see [9]).

3 Specifying games

In this section we consider the specification of games, based on observations and recursive modelling. Our approach extends Rao and Georgeff's translation of decision trees to $\text{BDI}_{\text{CTL}^*}$. We take their decision problems as our starting point.

3.1 Decision trees

Rao and Georgeff [8] use an extension of $\text{BDI}_{\text{CTL}^*}$ with probabilities and utilities to model single agent decision trees. This is done in the following way:

- Alternatives of decisions are modelled as branches in the branching time logic CTL.
- Beliefs of a decision maker model uncertainty. The translation of decision trees to $\text{BDI}_{\text{CTL}^*}$ models encodes all uncertainty as uncertainty about the actual world, such that given an actual world all actions are deterministic (a well known translation of indeterministic effects in decision theory).
- Goals of a decision maker are the alternatives the agent takes into consideration.
- Intentions of a decision maker are the optimal alternatives.

In this paper we remain faithful to the qualitative setting of BDI logic such that we do not use the probabilities and utilities. We represent the distinction between events and facts by introducing in the logical language a distinction between decision variables and parameters [7], also called controllable and uncontrollable propositions.decision variable in our approach corresponds to the proposition $done(e)$ in Rao and Georgeff's approach, where e is an event. We thus identify decisions with attributing true to decision variables. Joint actions like lifting table can be modelled by two individual decision variables a_1 and a_2 and believed consequences $a_1 \wedge a_2 \rightarrow p$, for p is lifting table.

Definition 4 *Assume n agents. The controllability of the variables is a partitioning of the propositional variables in $A_1, \ldots, A_n, P$, where A_i are controllable propositions of agent i, and P the propositions which are not directly controlled by a single agent.*

We keep the interpretation of belief, goal and intention worlds. The additional concept we introduce is that beliefs and goals about other agents are used to model the various layers of recursive decision models. Beliefs thus refer to other agents' decision models.

3.2 Games and recursive modelling

In our running example of a recursive decision problem, first agent 0, a legislator, decides which norm to create, then agent 1, a citizen, decides whether or not to fulfill this and other norms, then agent 2, a judge, decides whether or not the behavior of the citizen counts as a violation, and finally agent 3, a policeman, decides whether or not to sanction the behavior. In such recursive decision problems, each agent has to simulate the decision problem of each agent making a decision later than itself. The judge thus has to model the decision problem of the policeman, the citizen has to model the decision problem of the judge and the policeman, and the legislator has to model the decision making of the other three agents. Each of these decision models has to be solved a number of times. For example, the legislator has to solve the citizen's decision problem for each of its own

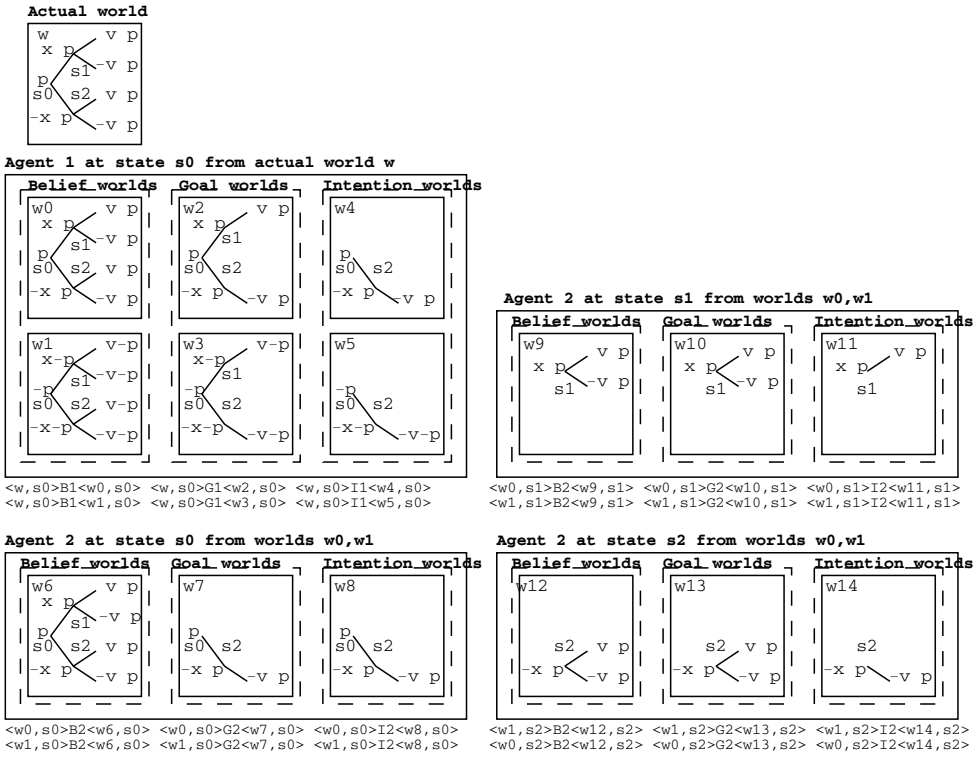


Figure 1: Recursive modelling.

decisions, it has to solve the decision problem of the judge for each possible decision of the legislator and the citizen, *et cetera*. Consequently, in recursive modelling an agent has to solve a large number of decision problems, before it can consider its own decisions. This leads to the problem of finding optimal decisions efficiently, the search for good heuristics, *et cetera*. However, for the specification problem considered in this paper, such efficiency considerations are less relevant.

In Figure 1 we illustrate the subgame between agent 1 which can play x or $\neg x$ and agent 2 which can reply with v or $\neg v$. Moreover, a parameter p represents different circumstances in the alternative worlds. Given the actual world w, the beliefs of agent 1 in state s_0 are represented by worlds w_0 and w_1 starting at s_0. Agent 1 considers that either x or $\neg x$, becasue w_2 and w_3 satisfy in s_0 $G_1(EXx)$ and $G_1(EX\neg x)$. It has the goal that x is followed by v, because this is the result of the recursive modelling of agent 2's behavior in w_{11}. Agent 1 may not like v, but this is not represented in the model.

Note that the intention attributed to agent 2 by agent 1 before its decision is different from after the decision. The initial goal of agent 2 in world w_7 is that x is false and it is followed by $\neg v$. However, after x in w_{10}, since x is observable by agent 2 (we have $B_1(AX(x \rightarrow B_2(x)))$), it changes its mind in world w_{11} and intends to reply with v.

3.3 Formal properties

The first assumption of recursive modelling is that we do not consider simultaneous or parallel decisions, because a recursive decision problem contains a sequence of agents who make a decision. In our formalization, such simultaneous decisions may be belief accessible because the decisions may be possible, but they are not goal accessible because they will not be taken into account. This is characterized by the following axiom. For each two distinct agents i and j, i.e., $i \neq j$, for each proposition d_1 built from A_i and d_2 built from A_j (such that d_1 and d_2 are not tautologies):

$$G_i d_1 \rightarrow G_j \neg d_2$$

The second assumption which we make here is that that each agent only makes a single decision in the decision problem. This is characterized by the following property for any two decision variables p_1 and p_2 of the same agent j, i.e., $p_1, p_2 \in A_j$, and for any sequence $AXAX \ldots$, which we write as $(AX)^n$.

$$G_i p_1 \rightarrow G_i (AX)^n \neg p_2 \text{ for } n > 0$$

To model such a sequence of decisions, we have to encode two things. First, the crucial mechanism for recursive modelling is that an agent assumes that each recursively modelled other agent's decisions are fixed for it. So, in our running example for agent 0 the decisions of agent 1, 2 and 3 are fixed for it, for agent 1 the decisions of agent 2 and 3 are fixed, and for agent 2 the decisions of agent 3 are fixed. The notion of fixation can now be characterized as follows. If $j > i$ and α is built from A_j, then:

$$B_i(I_j(EX\alpha)) \rightarrow G_i(EX\alpha) \qquad B_i(I_j(AX\alpha)) \rightarrow G_i(AX\alpha)$$

For example, in the model in Figure 1, we have due to the axioms above that $B_1(AX(x \rightarrow I_2(AXv))$, since $\langle w_{11}, s_1 \rangle$ is the only accessible situation from all the believed situations $\langle w_0, s_1 \rangle$ and $\langle w_1, s_1 \rangle$.

Second, when an agent is not making a decision, it may be influenced by the decisions of other agents. This is done by observing the decision or its effects. For example, the initial state of the agent 2's decision is its initial state, updated by observations from agent 1's decision. We do not formalize sensing actions.

Definition 5 *The set of observable atoms of agent i is a subset of the propositional variables. We say that $Ob_i(p)$ is true if p is built from observable atoms of agent i.*

The characteristic property of observations is that if an agent i decides something, then another agent j can observe it and assume it as fixed for its decision making.

$$Ob_j(p) \rightarrow B_i(AX(p \rightarrow B_j(p)))$$

The models may also satisfy a version of the above fixation axioms for beliefs, which is related to a notion of realism which can be adopted for other agents beliefs. We call this property transparency of decision variables: every agent considers possible the alternatives at disposal of the other agents. If α is built from A_j, then:

$$B_i(B_j(EX\alpha)) \rightarrow B_i(EX\alpha) \qquad B_i(B_j(AX\alpha)) \rightarrow B_i(AX\alpha)$$

For example, in Figure 1 agent 1 in w_0 and w_1 believes that the options of agent 2 are v and $\neg v$, because agent 1 believes that agent 2 that it has these options in w_9 and w_{12}.

4 Obligations and characterizing agents

We formalize obligations in this setting. It is inspired by the so-called Anderson's reduction of deontic logic to alethic logic [1], which may be written as $O(p) = \Box(\neg p \rightarrow V)$. In this definition V is a so called violation constant. To distinguish between violations, we assume that there is a set of them. In particular, we assume that a subset of the propositions represents that a norm of this normative system is violated, see [11] for details.

Definition 6 *The violation set is a subset of the propositional variables. We say that $V(p)$ is true if p is built from the violation set only.*

Likewise, some decision variables are known as sanctions.

Definition 7 *The sanction set is for each agent a subset of the propositional variables. We say that $S_i(p)$ is true if p is built from the sanction set of agent i.*

There are various ways to encode obligations in a BDI setting. Following ideas in [2, 3, 4] we say that an obligation of an agent corresponds to a goal of the normative system. This has been paraphrased by "your wish is my command". Judges and policemen are defined by further obligations. Due to space limitations, we cannot discuss this definition any further.

Definition 8 (Obligations) *Agent i is* obliged *to do a (a decision variable in A_i) with sanction s, represented by $O_{i,j,k,l}(a, s)$, iff there exists agents j, k, l such that:*

1. *$G_j(a)$: agent j (legislator) wants that a;*
2. *$G_j(\neg a \rightarrow AXp)$: agent j wants that absence of a implies p;*
3. *$p \in A_k$ and $V(v)$: v is a decision variable of agent k (judge) which says that behavior counts as a violation;*
4. *$G_jAX(v \rightarrow AXs)$: agent j wants that v implies s;*
5. *$s \in A_l$ and $S_i(s)$: s is a decision variable of agent l (policeman) which represents a sanction for agent i;*
6. *$G_jA\Box\neg v$: agent j desires that there are no violations.*
7. *$G_jA\Box\neg s$: agent j desires that there are no sanctions.*

The spirit of the Montesquieu's *trias politica* is the principle of the separation of powers. Autonomous agent i is respectively:

legislator if it can change the obligations, i.e., there is some $1 \leq a \leq n$ such that we have $(\neg O_{a,i,k,l}(p, s) \wedge a) \rightarrow AXO_{a,i,k,l}(p, s)$.

judge if it has the power to count behavior of citizens as a violation, i.e., there is a $a \in A_i$ such that $V(a)$.

policeman if it can sanction behavior, i.e., there is a $a \in A_i$ such that $S_j(a)$ for some j.

The separation of powers in the trias politica means that agents cannot be both a legislator and a judge, or both a legislator and a policeman, or both a judge and a policeman. This can be characterized in the obvious way.

5 Concluding remarks

In this paper we formalize the specification of games in the trias politica. We use Rao and Georgeff's specification language $\text{BDI}_{\text{CTL}^*}$. In particular, we generalize Rao and Georgeff's specification of single agent decision trees to multiagent games, for which we introduce observations and recursive modelling. We formalize obligations in this setting, and we distinguish four kinds of agents, called legislators, judges, policemen and citizens. Legislators are characterized as agents that can change the obligations, judges are characterized by their power to count behavior of citizens as a violation, and policemen are characterized by their ability to sanction behavior.

Each set of normative agents can again be ordered in higher and lower order legislators, judges and policemen, which leads to three hierarchies of normative agents. A higher court considers which permissions it can create such that lower courts will not introduce norms the higher court deems undesirable [5]. The specification of such games is subject of further research.

References

[1] A. Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, 67:100–103, 1958.

[2] G. Boella and L. Lesmo. A game theoretic approach to norms. *Cognitive Science Quarterly*, 2(3-4):492–512, 2002.

[3] G. Boella and L. van der Torre. Attributing mental attitudes to normative systems. In *Procs. of AAMAS'03*, Melbourne, 2003. ACM Press.

[4] G. Boella and L. van der Torre. Rational norm creation: attributing mental attitudes to normative systems, part 2. In *Procs. of ICAIL 03*, pages 81–82, Edinburgh, 2003. ACM Press.

[5] G. Boella and L. van der Torre. Permissions and obligations in hierarchical normative systems. In *Procs. of ICAIL 03*, pages 109–118, Edinburgh, 2003. ACM Press.

[6] P. J. Gmytrasiewicz and E. H. Durfee. Formalization of recursive modeling. In *Procs. of ICMAS-95*, 1995.

[7] J. Lang, L. van der Torre, and E. Weydert. Utilitarian desires. *Autonomous Agents and Multiagent Systems*, pages 329–363, 2002.

[8] A. S. Rao and M. P. Georgeff. Deliberation and its role in the formation of intentions. In *Procs. UAI'1991*, pages 300–307, San Mateo, CA, 1991. Morgan Kaufmann Publishers.

[9] A. S. Rao and M. P. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–343, 1998.

[10] K. Schild. On the relationship between BDI logics and standard logics of concurrency. *Autonomous Agents and Multi-Agent Systems*, 3(3):259–283, 2000.

[11] L. van der Torre and Y. Tan. Diagnosis and decision making in normative reasoning. *Artificial Intelligence and Law*, 7(1):51–67, 1999.

[12] M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

A Test Bed for Multi-Agent Systems and Road Traffic Management

Alexander Th. van den Bosch [a] Maarten R. Menken [a]

Martijn van Breukelen [b] Ronald T. van Katwijk [c]

[a] Vrije Universiteit, Amsterdam, {atvdbosc, mrmenken}@cs.vu.nl
[b] TNO TPD, Delft, breukelen@tpd.tno.nl
[c] TNO Inro, Delft, r.t.van.katwijk@inro.tno.nl

Abstract

In this paper we present a test bed for multi-agent systems in road traffic management. This software environment is developed at TNO to aid in-depth research in this field. The test bed establishes a connection between the traffic simulator Paramics and the multi-agent platform JADE. Special Jess agents have been developed to accelerate the implementation of different agent-based traffic control strategies. Two example implementations demonstrate the functionality of the developed system and function as a starting point for further research.

1 Introduction

Throughout the world, traffic congestion forms a daily recurring problem. In The Netherlands, more and more instruments are deployed to stimulate an undisturbed flow of traffic on highways and urban road networks. For example, ramp metering installations (RMIs) are used to regulate the inflow of traffic at on-ramps. Another example is the variable message sign (VMS), which keep motorists informed about the current road conditions ahead. They can also realize temporary lane blockings and overrule maximum speeds. Research is being conducted in the field of automatic coordination of these dynamic traffic management instruments. Van Katwijk and Van Koningsbruggen argue in [15] that the communication capabilities of multi-agent systems can be used to accomplish this coordination.

In the project 'Verkeerscentrale van de Toekomst', TNO aims at the development of new concepts for the changing role of traffic operators. In one of the concepts, multi-agent systems are used to assist the real-time coordination of automatic traffic management instruments. The task of the human traffic operator is shifted towards higher-level traffic control.

Yet, no consensus exists about the best configuration of the traffic managing multi-agent system. In the vision of TNO, the system should be capable of managing different levels of complexity, a diversity of policy goals, and different forms of traffic problems.

To be able to experiment with different strategies for the application of multi-agent systems for dynamic traffic management and to examine their applicability, TNO formulated the need for a test bed. The test bed should facilitate the development of multi-agent systems for dynamic traffic management. It should be easy to use and test multi-agent strategies in a realistic simulated traffic environment and the development process should not be hindered by cumbersome implementation details. This paper describes an implementation of this test bed that was developed during an internship at TNO, conducted by the first two authors. It also describes the two experiments that have been conducted in order to demonstrate its functionality. For a detailed discussion of the material, we refer to [13], on which this paper is based.

2 Related Work

Several technologies from the field of Artificial Intelligence are being applied in dynamic traffic management including Evolutionary Algorithms, Knowledge-Based Systems, Neural Networks and Multi-Agent Systems [12]. For clarity, a distinction can be made between vehicle-oriented and measure-oriented traffic control:

Vehicle-oriented traffic control focuses on the control of the behavior of individual vehicles. For example, Moriarty, Handley and Langley model each vehicle as an intelligent agent that can be influenced [10]. Adler and Blue present a traffic control method through in-vehicle routing and navigation systems in [6]. Other examples are provided by [5] and [1].

Measure-oriented traffic control assumes that traffic consists of entities with goal-oriented behavior. Control of this behavior is performed through external signals at fixed locations, like traffic lights and variable message signs. Our research focuses on this form of control.

To research the role of multi-agent systems in measure-oriented traffic management, Hernández, Cuena and Molina have developed the TRYS model [9]. So-called 'problem areas' are defined in a particular traffic situation. Each problem area has an agent assigned to it. The agents formulate actions to be performed and propose them to a 'coordinator', who makes a final decision in case of conflicting plans. Roozemond and Rogier also propose higher-level agents to perform conflict resolution [11]. The authors list adaptability, communication and proactive behavior as the most important benefits of multi-agent systems in traffic control.

Ferreira, Subrahmanian and Manstetten take another approach by presenting a fully decentralized traffic control mechanism using agent technology in [8]. No higher-level agents are present for conflict resolution. The sharing of 'opinions', numerical estimations of the traffic situation in particular areas, must result in conflict-minimizing multi-agent behavior. Simulations have showed that the use of these opinions can result in coordinated and more efficient traffic control.

Van Katwijk and Van Koningsbruggen propose in [15] the use of multi-agent systems to relieve the task of the human traffic operator. They demonstrate by

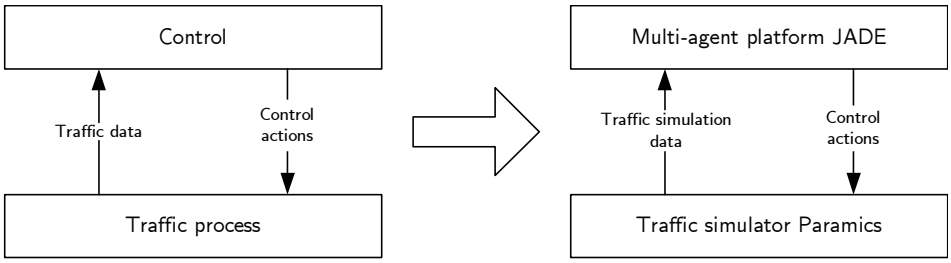


Figure 1: Traffic management cycle.

means of two examples the need for coordination of traffic control measures and the benefits of multi-agent systems in this field. According to the authors, the main advantages of multi-agent systems are proactiveness and social behavior. A hierarchical multi-agent structure for the sake of conflict resolution is suggested. Van der Arend [14] was the first at TNO to actually implement a multi-agent system for dynamic traffic control. The new ideas and research questions that resulted from this first implementation, and the ambition of TNO to participate in research on this topic, gave rise to the desire for the test bed presented here.

3 Software Components of the Test Bed

In traffic control, two processes can be distinguished, as depicted in Figure 1. First of all, there's the traffic process in which all motorists and other traffic participants are involved. Through the use of instruments the traffic situation can be observed. These observations are used in the traffic control process. In this second process, decisions are made when and how to perform dynamic traffic management measures in order to optimize the traffic process.

In our test bed the traffic process is simulated by Paramics; the control is delegated to a multi-agent system implemented in JADE. This way a mapping can be made from the actual traffic management cycle to a simulated environment. This is shown on the right-hand side of Figure 1.

This section discusses how the test bed simulates the traffic process and can be used to build implementations of multi-agent strategies for the traffic control process. Paramics, JADE and two other important software components that make up the test bed are presented in the next subsections.

3.1 Paramics

Paramics is a microscopic traffic simulation suite developed by Quadstone Ltd. [3]. Figure 2 contains a screen shot of the Paramics Modeller. The Modeller is used to define a traffic network and the amount of simulated traffic. After that, the traffic process is simulated on the level of individual vehicles. Paramics can be extended through an API. User-defined plug-ins, written in C, can retrieve traffic simulation information from Paramics and send back control actions.

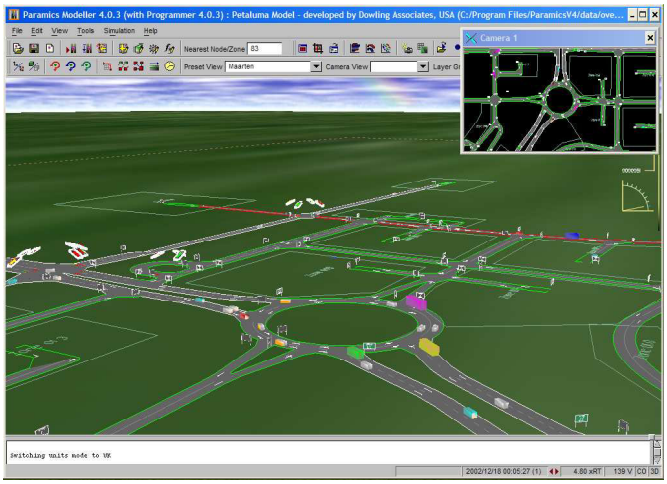


Figure 2: Screen shot of the Paramics Modeller.

3.2 JADE

JADE is an agent development environment implemented in the Java language [7]. It provides the facilities for agent autonomy, inter-agent communication, task execution and agent management. The agent developer extends the JADE Agent and Behaviour classes to develop application-specific agents and tasks.

3.3 Paramics-JADE Software Bridge

The third component of the test bed is the Paramics-JADE software bridge. This interface has been developed both to be able to provide agents with traffic information from the Paramics simulation and to allow agents to control the simulated traffic measures such as ramp metering installations (RMIs). The bridge has been implemented as a plug-in for Paramics using both the C and Java programming languages.

3.4 Generic Jess Agent

For our test bed we developed a generic Jess agent of which the agent-specific task is executed by Jess, a Java rule-based reasoning engine [2]. Incoming messages are converted to Jess facts and asserted into its working memory. Derived facts describing messages to be sent are translated into corresponding JADE messages, after which JADE takes care of their delivery. This Jess agent has a number of advantages.

- It enables the user to make use of the pattern-matching capabilities of Jess. As a result, the user can focus on the logical aspects of the agent's knowledge.

- An agent window displays the state of the agent and gives insight in internal agent processes. This includes its knowledge about the world and other agents, and a history of its derivations. This way the reasoning process can be traced easily.
- With the generic Jess agent, different agents can be defined without the need to compile the specific instances. Agents are defined by providing an instance of the generic Jess agent with agent-specific knowledge in Jess format. When the agent is created in JADE, this knowledge is loaded into the rule-engine.

From the perspective of the multi-agent system developer, the following two phases can be distinguished:

1. Configuration phase. The user creates the traffic situation to be simulated and the interacting multi-agent system.
2. Simulation phase. The simulation is started, simulation data is collected and control actions are derived and communicated by the multi-agent system.

A special XML file lists the agents that make up the multi-agent system. Adding an agent is done by adding a line containing its name and the location of the agent-specific knowledge. The next step is to actually construct the Jess files containing the knowledge. When the simulation is started, the XML file is parsed, the listed agents are created in JADE, and they are provided with their corresponding agent-specific knowledge.

4 Scenario

To demonstrate the test bed, we implemented two alternative multi-agent systems for the first example described in [15] by Van Katwijk and Van Koningsbruggen. Figure 3 depicts the traffic situation, containing one highway and three entrance ramps. The inflow from each ramp to the highway can be regulated by a ramp metering installation. Suppose congestion starts to build up on the highway downstream of the third ramp. In this case, the third RMI will detect this and will reduce its inflow. The second RMI will not start to do the same until the congestion has reached the area directly downstream of the second entrance point. This also applies to the first metering installation.

Using our software, we have developed a multi-agent system, capable of communicating about the actions to be performed. As a result of this communication, the ramp metering installations are better aware of future congestion. In this way, they can react in advance and solve the congestion timelier.

In our first agent implementation the highway is divided into three parts, each provided with two detectors and one agent. Based on the detector data the agent forms an image of the current traffic situation. Depending on this estimate and on incoming messages from agents downstream, the state 'no problems', 'request for help, urgency low' or 'request for help, urgency high' is communicated to the neighboring agents upstream. Apart from these three highway agents, three RMI

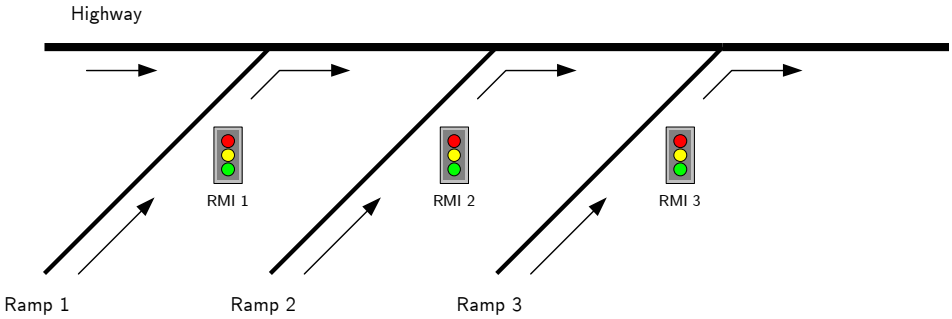


Figure 3: Traffic network used in the scenario.

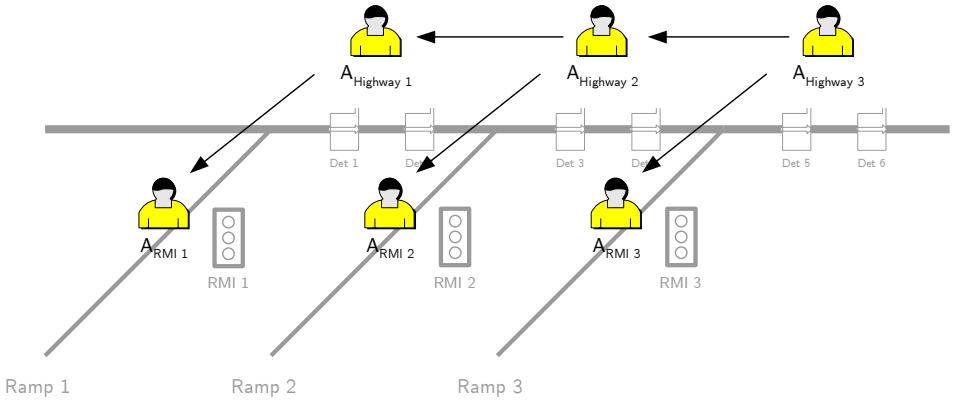


Figure 4: Coordination through infrastructure agents.

agents are defined. Their only task is to control the RMI, based on incoming help requests. They are not responsible for the traffic situation on the ramps. This multi-agent configuration is shown in Figure 4. The arrows indicate the communication possibilities of each agent. In this way the knowledge about the current traffic situation propagates upstream through the network.

Figure 5 shows an example of a Jess rule. It is part of the rule base of the first highway agent and describes the situation in which congestion occurs near Detector 2 and no problems are reported by the second highway agent. In this case, the first highway agent decides to send a request for help to the first RMI agent.

In the agent implementation of Figure 4 messages travel along the infrastructure. Another way to arrange the communication is to make the instruments responsible for the communication upstream. In that case the arrows between the highway agents are moved downwards to the RMI agents. Simulations show that in both agent implementations congestion is tackled earlier. This can also be demonstrated by means of traces, which are elaborated in [13, pp. 65–68].

The two agent implementations proved that the system enables users to create

```
(defrule minor-problems
    (simulation-time ?time)
    (average-speed ?time "Detector2" ?average-speed-downstream)
    (average-speed ?time "Detector1" ?average-speed-upstream)
    (test (<= ?average-speed-downstream 22.2))  ; 22.2 m/s = 80 km/h
    (test (> ?average-speed-upstream 22.2))
    (state ?time "HighwayAgent2" "no-problems")
  =>
    (assert (ACLMessage (sender "HighwayAgent1")
                        (receiver "RMIAgent1")
                        (performative inform)
                        (content "request-for-help, urgency-low")))
)
```

Figure 5: Jess rule of the first highway agent (adapted).

different multi-agent systems for dynamic traffic management. Furthermore, the development process is simplified and accelerated by the incorporation of rule-based reasoning.

5 Conclusions and Future Work

To aid the ongoing research in this field, we developed a software environment for rapid development of multi-agent systems able to interact with a traffic simulation. In this paper a test bed for agent-based road traffic management is presented. The organization of the software is discussed, as well as the role of rule-based reasoning. The implementation of two multi-agent systems with the test bed is described. The two existing implementations show that multi-agent systems can be created easily.

In our opinion the presented test bed will be of great value for developments in traffic management. However, the developed system has opportunities for further extension. A graphical user interface can be developed in which agents can be created with only a few mouse-clicks and in which agent communication is visualized. This would further accelerate the implementation of the desired multi-agent system. Extending the number of available traffic control instruments could be another improvement.

With the test bed, a tool has been developed to study the possibilities of applying multi-agent systems in dynamic traffic management. The presented agent implementations can be a starting point for further research. For example, the mechanism can be extended with agents who are responsible for traffic on the ramps. Negotiation can be a means of weighing the flow on the highway against the interests of vehicles on the entrance ramp.

References

[1] Dynamic vehicle routing using ant based control. Website, May 2002. URL: http://www.kbs.twi.tudelft.nl/Publications/MSc/2002-Kroon-MSc.html.

[2] Jess, the Expert System Shell for the Java Platform. Website, May 2003. URL: http://herzberg.ca.sandia.gov/jess/.

[3] Paramics: Microscopic Traffic Simulation developed by Quadstone Ltd. Website, May 2003. URL: http://www.paramics-online.com/.

[4] TNO Verkeer en Vervoer. Website, augustus 2003. URL: http://www.vv.tno.nl/.

[5] Website AIDA. Website, juli 2003. URL: http://www.aida.utwente.nl/.

[6] Jeffrey L. Adler and Victor J. Blue. Principled Negotiation and Multiagent Transportation Management an Information Systems. presented at the Transportation Research Board 80th Annual Meeting, January 2000.

[7] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *JADE PROGRAMMER'S GUIDE*. TILab S.p.A., July 2002. URL: http://sharon.cselt.it/projects/jade/.

[8] Enrique D. Ferreira, Eswaran Subrahmanian, and Dietrich Manstetten. Intelligent agents in decentralized traffic control. In *IEEE Intelligent Transportation Systems Conference Proceedings*, pages 707–711, Oakland (CA), USA, August 2001. IEEE.

[9] Josefa Hernández, José Cuena, and Martín Molina. Real-time Traffic Management through Knowledge-based Models: The TRYS approach. In ERUDIT, editor, *ERUDIT Tutorial on Intelligent Traffic Management Models*, Helsinki, Finland, 1999.

[10] David E. Moriarty, Simon Handley, and Pat Langley. Learning Distributed Strategies for Traffic Control. In *Proceedings of the Fifth International Conference of the Society for Adaptive Behavior*, Zurich, Zwitserland, 1998.

[11] Danko A. Roozemond and Jan L. H. Rogier. Agent Controlled Traffic Lights. In *ESIT 2000; European Symposium on Intelligent Techniques*, Aachen, Duitsland, September 2000.

[12] Danko A. Roozemond, Henk J. van Zuylen, and Peter van der Veer. Self-Regulating Urban Traffic Control Systems. In *WCTR; World Conference on Transport Research*, Seoul, July 2001.

[13] Alexander Th. van den Bosch and Maarten R. Menken. Multi-Agentsystemen en Verkeersbeheersing - De Ontwikkeling en Demonstratie van een Testbedomgeving. Technical Report 03-7N-109, TNO, Delft, mei 2003. Afstudeerscriptie, Vrije Universiteit, Amsterdam.

[14] R. J. M. van der Arend. Intelligente Agents in Verkeersbeheersing - De ontwikkeling van theoretische agentstructuur en een praktische implementatie en simulatie daarvan. Report 02-7N-112-72873, TNO Inro, Delft, maart 2002.

[15] Ronald T. van Katwijk and Paul van Koningsbruggen. Coordination of traffic management instruments using agent technology. *Transportation Research Part C: Emerging Technologies*, 10(5-6):455–471, October–December 2002.

MIRA: a Medical Information Retrieval Agent

Loes Braun[a] Floris Wiesman[a] Jaap van den Herik[a]

Arie Hasman[b]

[a] Institute for Knowledge and Agent Technology, Universiteit Maastricht, P.O.Box 616, 6200 MD Maastricht

[b] Department of Medical Informatics, Universiteit Maastricht, P.O.Box 616, 6200 MD Maastricht

Abstract

Physicians have to make a multitude of decisions, some of which cannot be made without using information from a variety of information sources. If physicians are provided with relevant information from these sources, it is probable that the quality of care will be improved and the length of stay of patients will be reduced. Because quality of care and length of stay are important parameters in health care, the long-term goal of our research is the development of a system, MIRA, which provides the physicians with relevant and patient-specific medical information. The short-term goal of our research is to identify the problems that will occur when MIRA (a) extracts a physician's information needs based on specific patient data, (b) formulates queries based on the information needs, and (c) executes the queries on a medical literature database. Here we encountered four problems: (1) information-need extraction, (2) terminology mapping, (3) query formulation, and (4) results presentation. We investigated these problems with the help of a prototype and we may conclude that we have found potential solutions to the problems, some of which are partly implemented in the MIRA prototype.

1 Medical Literature in Context

Physicians are typically facing a multitude of decisions. Some of these decisions cannot be made without using information which is hidden in a variety of information sources. Owing to the rapid growth of the size and number of the available information sources, the amount of time needed to retrieve relevant information from the information sources increases considerably. Since physicians work under extreme time pressure with only little opportunities to browse medical information sources, the retrieval of relevant information is a serious problem.

Some years ago, Gamble [3] already showed that the retrieval of relevant literature about specific patients is vital to the quality of care. However, physicians lack the time to retrieve this literature [3]. Therefore, the quality of care is still not optimal. To improve this situation, a system is required that retrieves relevant literature without disturbing the physician. To be independent of the intervention

of a physician, an autonomous system should have access to the patient context (for the formulation of search requests and the presentation of the search results). In the long term our research aims at the development of such a system: MIRA (Medical Information Retrieval Agent). In the short term, the goal of our research is to identify the various problems that should be solved before building MIRA and to indicate potential solutions to these problems.

MIRA is part of the MIA (Medical Information Agent) project.[1] This project aims at the development of a system which supports physicians by providing them with (a) feedback about their actions, (b) advice concerning their course of action, and (c) patient-specific medical literature.

The remainder of the article is as follows. Section 2 discusses related work. Section 3 describes the identified problems and their solutions and Section 4 provides conclusions and directions for future research.

2 Related Work

Medical information retrieval in context has been the topic of several research projects. For brevity, we only provide the original ideas of two early projects [4, 7] and complete this with a more recent project [9].

Miller et al. developed a prototype that links a diagnostic support system to a bibliographic search system [4]. Despite the fact that the system provides the user with 'canned' search logic, the user had to specify (a) the disease, (b) additional Medical Subject Headings (MeSH) [5] if desired, and (c) various search parameters.

Rada et al. developed the OAR (Open Architecture for Reasoning) prototype to switch between a patient's electronic medical record (EMR) and related medical literature [7]. To enable the system to formulate appropriate queries, the physician had to select relevant patient data and a type of query.

Van Mulligen developed a system that enabled physicians to select patient data which they deemed important from an EMR. The system them used these data to search in medical literature databases for relevant literature [9].

The overall short-coming of these systems is that the degree of necessary interaction of the physician with the system is too high. Consequently, physicians will be reluctant to use the system as part of their daily routine.

3 MIRA: Medical Information Retrieval Agent

The main problems in medical information retrieval are the lack of time of physicians and the time-consuming nature of information retrieval. A physician's retrieval process consists of five time-consuming steps.

- Formulation of information needs,
- Translation of the information needs into the terminology used by the database on which information retrieval is performed,

[1]The MIA project is funded by NWO (project number 634.000.021).

- Formulation of queries,
- Retrieval of documents, and
- Inspection of the search results.

In sections 3.1 to 3.5 we discuss each step separately. For each step we identify the problems that have to be solved to automate the step. Moreover, we indicate potential solutions to these problems, which are implemented into a prototype of MIRA. In section 3.6 the architecture of the MIRA prototype is described.

3.1 Formulation of Information Needs

The first step in the retrieval process is the formulation of one or more of the physician's information needs. To minimize the physician's time and interaction needed to perform this step MIRA has to determine the information needs in a proactive and autonomous way. This is a difficult task, since the information needs are implicit and even the physician himself may not be aware of them. Hence, the determination of information needs constitutes our first problem. A potential solution, implemented into the MIRA prototype, is to use knowledge about physicians' information seeking behaviour and to use the patient data from the EMR as context. From [2] it was clear that the information needs of physicians often have a similar structure and differ only in the content. Consequently, these information needs can be modelled into generic forms. We will refer to such forms as information-need templates. The use of information-need templates raises a subproblem, for it is difficult to determine what information-need templates to use. A potential solution to this subproblem, used in MIRA's development, is to consult literature (e.g., [8]) on what information needs are common. Within MIRA a small set of information-need templates is implemented. Obviously, to make MIRA usable in practice, the set of templates should be expanded. From [8] we found a sample information need which we modelled into the information-need template:

'To diagnose (exclude or confirm) of <DISEASE>*, how good is* <DIAGNOSTIC PROCEDURE> *for this patient?'*

During the retrieval process, the templates are instantiated with the patient data from the EMR. A second subproblem is to determine which information-need templates should be instantiated. One solution, implemented in MIRA, is to base the instantiation on the stage of the patient's treatment. MIRA determines this stage from the EMR. MIRA can then determine which information-need templates should be instantiated. For example, if an EMR contains the diagnosis for the specific patient, but no information about the medication corresponding to this diagnosis is available, the physician will probably prefer information on medication over information on diagnosis. Therefore, templates concerning diagnosis will not be instantiated automatically, but templates concerning medication will. In the final version of MIRA an agent will be used to detect changes in the EMR and to monitor the various stages of the treatment. When there is a change in the EMR the agent

will trigger a new retrieval proces based on the new information and the stage of the treatment. Since it is possible that the quality of the resulting documents is not sufficiently high, the final version of MIRA will also enable physicians to select information-need templates which they think should be instantiated. In this way, the physician himself controls the degree of interaction and the search results might be improved.

To achieve correct instantiations, information-need templates should only be instantiated with data of the type needed in the template. Determining the type of the data is difficult, since the data may use a particular terminology or, in case of narrative text, no terminology at all. Therefore, a third subproblem was the determination of the various information types within an EMR. A potential solution to this problem, implemented in the MIRA prototype, is to annotate the EMR data with information types. Examples of information types which are also used in the above-mentioned information-need template are: <DISEASE> and <DIAGNOSTIC PROCEDURE>. After the annotation, MIRA is able to determine whether a part of the patient data is, for example, a disease or a diagnostic procedure. To instantiate an information-need template, the prototype determines the information types in the template (e.g., <DISEASE> and <DIAGNOSTIC PROCEDURE>). It then searches the complete patient record for information of this type. It then instantiates the template for each occurrence (e.g., *'For diagnosis of deep vein thrombosis, how good is ultrasound?'*). To make MIRA usable in practice, annotating the EMR with information types should not be a necessary item. It can be averted by using a controlled vocabulary when entering the data into the EMR. Physicians have to fill out each entry in the EMR by choosing from a list of options, which are linked to the Unified Medical Language System (UMLS) [6]. Via UMLS MIRA will be able to determine to which information type the entry belongs.

3.2 Translation of the Information Needs

The second step in the retrieval process is to translate the extracted information needs into the terminology used by the medical literature database on which retrieval will be performed. EMRs (from which the information needs are extracted) and medical databases use different terminologies. For instance, EMRs often use ICD10 (International Classification of Diseases), HL7 (Health Level 7), or SNOMED (Systemized Medical Nomenclature), whereas medical databases often use MeSH. To ensure a good retrieval performance the information needs and the database should use the same terminology. Therefore our second problem is to map the various different medical terminologies to each other.

For the development of MIRA's prototype we started with retrieving literature from only one specific database: the Cochrane Library [1], containing about 400,000 documents (in August 2003). We chose this database for two reasons. First, the Cochrane Library contains metastudies, and hence provides more general conclusions to the user than regular medical literature. Second, the articles in the Cochrane Library are clearly structured, which makes them easy to read for the user. A disadvantage of the Cochrane Library is that its articles are less up to date than the articles in regular medical literature databases.

A potential solution to the terminology-mapping problem, implemented in the MIRA prototype, is to annotate the data in the EMR with MeSH terms. This solution was based on the use of the Cochrane Library, since this database uses MeSH too. By using the annotated terms instead of the real data in the EMR, the information needs (extracted from the EMR) will also use the MeSH terminology.

This solution is not optimal for two reasons. First, annotating the EMR data is quite a time-consuming task. Second, databases other than the Cochrane Library use different terminologies and new terminologies might become available. Therefore, in the final version of MIRA a controlled vocabulary (discussed in subsection 3.1) will be used instead of annotation. Via UMLS the terminologies can be mapped.

3.3 Formulation of Queries

The third step in the retrieval process is the formulation of queries. The query formulation influences the number of documents in the results set and, consequently, the physician's time needed to inspect this set. Therefore, the query formulation is our third problem. To minimize the time needed for the inspection of the results, it is important that as few non-relevant results as possible are be retrieved. This implies that the result set should have a high precision (i.e., the number of non-relevant documents retrieved is as small as possible). High-precision result sets often have a low recall (i.e., the percentage of relevant documents which is retrieved is low). However, for our purpose, a low recall is no disadvantage, because it is not important to the physician to find *all* relevant documents about a specific topic. A single document that answers his question is sufficient. To achieve results with a high precision, the detail level of the queries is very important. Determining the optimal detail level is the first subproblem of the query-formulation problem. The detail level is dependent on the specific query language used. Since most medical literature databases (including the Cochrane Library) use a Boolean query language, the precision of a query result will be as high as possible if the query is highly detailed. To determine the optimal detail level we performed an informal evaluation of various queries (enhanced with various data from the EMR) based on information-need templates used in MIRA's prototype. Assume we use the EMR of a 51 year old woman. No definitive diagnosis is available yet, but she is suspected to suffer from deep vein thrombosis and the physician asked for an ultrasound. From the EMR we formulated and tested the queries below.

1. diagnosis AND (evaluation NEXT procedure) AND ultrasound AND (deep NEXT vein NEXT thrombosis)

2. diagnosis AND (evaluation NEXT procedure) AND ultrasound AND (deep NEXT vein NEXT thrombosis) AND female AND adult

We retrieved documents for each of the two formulations and computed the comparative recall and precision for each formulation. We use 'comparative' recall, since for computation of the 'absolute' recall it is necessary to identify all relevant documents, which was not possible. The results are shown in Table 1. These

Table 1: Comparative recall (CR), precision (P), number of documents retrieved (DR), and number of relevant documents (RD) for two queries with a different detail level(Q).

Q	CR	P	DR	RD
1	1	0.75	4	3
2	0.33	1	1	1

results show that the precision is optimal for the second query, which was enhanced with information about the age and sex of the patient. Moreover, the retrieved document contained a clear answer to the question asked: *'Impedance plethysmography, venoscan, and ultrasound had accuracies of 65%, 80%, and 82%, respectively. [...] Of those tested, the X-ray venogram was the only investigation suitable for definitive diagnosis.'* We formulated and tested several queries, with different detail levels, based on each information-need template implemented in MIRA. From the results it transpired that the precision was optimal for queries which were enhanced with as much details as possible. Obviously, it is difficult to determine which data are relevant to enhanced a query. This constitutes a second subproblem of the query formulation problem.

3.4 Retrieval of Documents

The retrieval of documents is not a problem, because the actual retrieval process of MIRA is completely guided by the search engine of the Cochrane Library. The prototype passes a query to be executed to the Cochrane's search engine. The search engine retrieves the appropriate documents and passes them back to MIRA. Since MIRA itself does not retrieve the documents, it has no direct influence on the retrieval performance.

3.5 Inspection of the Results

The inspection of the results is the only step in MIRA's retrieval process in which it is always necessary for the physician to interact with the system. This interaction should be as unobtrusive as possible. Therefore, our fourth problem is the presentation of the results. The way in which the results are presented determines the effort a physician has to make to find relevant results from the complete set of results. Therefore, the first subproblem of the presentation problem is to present the results as conveniently arranged as possible. In MIRA's prototype the results are ordered according to the query which they correspond to. In the final version of MIRA the results will be ordered according to their relevance and they will be integrated into the EMR. Moreover, in the final version of MIRA, passage retrieval should be incorporated to decrease further the effort needed to find relevant results. To determine which passages are relevant is a difficult task and constitutes a second subproblem of the presentation problem.

3.6 System Architecture

In summary, we have found four problems:

1. Extraction of information needs
2. Mapping of terminologies
3. Optimizing the detail level of the query
4. Presenting the results in a convenient way

For each of these problems, a separate module was implemented. An additional module was implemented for the retrieval task. These modules were combined into the MIRA prototype. A schematic representation of the architecture of the prototype is given in Figure 1.

4 Conclusions

From our research, we may conclude that the development of a medical literature retrieval system comprises four problems: (1) extraction of the physician's

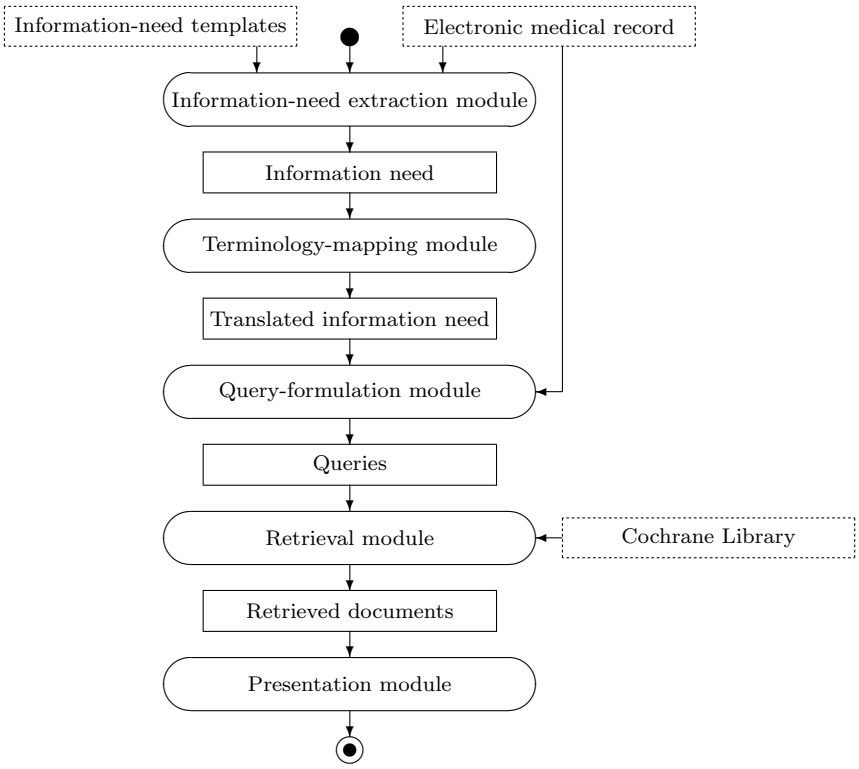


Figure 1: Schematic representation of the architecture of MIRA's prototype.

information need, (2) terminology mapping, (3) query formulation, and (4) results presentation. Moreover, we may conclude that we have found potential solutions to the four problems, some of which are partly implemented by the MIRA prototype.

There are several directions for future research. We mention one for each problem. The first direction is to generate a more extensive set of information-need templates and to enable physicians to add new information-need templates. The second direction is to facilitate terminology mapping via UMLS. The third direction is to perform extensive research on query formulation. The fourth direction is to integrate the search results into to EMR, to incorporate passage retrieval into the system, and to let physicians evaluate the system.

References

[1] The Cochrane Collaboration. The Cochrane Library 2003, issue 2, 2003.

[2] J.W. Ely, J.A. Osheroff, and M.H. Ebell. Analysis of questions asked by family doctors regarding patient care. *British Medical Journal*, 319(7206):358–361, 1999.

[3] S. Gamble. Hospital libraries enhance patient care and save money. *Journal of the Alberta Association of Library Technicians*, 23(2):10–12, 1996.

[4] R.A. Miller, L. Jamnback, N.B. Guise, and F.E. Masarie. Extending the capabilities of diagnostic decision support programs through links to bibliographic searching: Addition of 'canned mesh logic' to the quick medical reference (QMR) program for use with grateful med. In *Proceedings of the Fifteenth Annual Symposium on Computer Applications in Medical Care*, pages 150–155, 1991.

[5] U.S. National Library of Medicine. Medical Subject Headings 2003, 2003. http://www.nlm.nih.gov/mesh/meshhome.html.

[6] U.S. National Library of Medicine. Unified Medical Language System, 2003. http://www.nlm.nih.gov/research/umls/.

[7] R. Rada, J. Barlow, D. Bijstra, J. Potharst, P. de Vries Robbe, and P. Zanstra. OAR: Open architecture for reasoning applied to connecting patient records to medical literature. In J. Noothoven van Goor and J.P. Christensen, editors, *Advances in Medical Informatics: Results of AIM Exploratory Action*, pages 287–294, Amsterdam, The Netherlands, 1992. IOS Press.

[8] R. Smith. What clinical information do doctors need? *British Medical Journal*, 313(7064):1062–1068, 1996.

[9] E.M. van Mulligen. UMLS-based access to CPR data. *International Journal of Medical Informatics*, 53(1):125–131, 1999.

Modeling human color categorization: Color discrimination and color memory

E.L. van den Broek [a] M.A. Hendriks [a] M.J.H. Puts [b]
L.G. Vuurpijl [a]

[a] Nijmegen Institute for Cognition and Information,
P.O.Box 9104 6500HE Nijmegen
[b] Visual Science Laboratories, University of Chicago,
940 East 57th Street, Chicago, IL 60637

Abstract

Color matching in Content-Based Image Retrieval is done using a color space and measuring distances between colors. Such an approach yields non-intuitive results for the user. We introduce color categories (or focal colors), determine that they are valid, and use them in two experiments. The experiments conducted prove the difference between color categorization by the cognitive processes color discrimination and color memory. In addition, they yield a Color Look-Up Table, which can improve color matching, that can be seen as a model for human color matching.

1 Introduction

The origin of the color *lilac* lays in the Sanskrit *nilla 'dark blue'*, of which the Persian made *nIlak 'bluish'*, from *nIl 'blue'*. In the Arabic the meaning evolved to a description of a plant with flowers of this color: the *Sering*. In 1560 the *Sering* was brought to Vienna, by an Austrian ambassador. From there the plant reached France. There the word's meaning evolved to *"a variable color averaging a moderate purple"*[1].

So, there is more with colors than one would think at a first glance. The influence of color in our everyday life and the ease humans use color are in strong contrast with the complexity of the phenomenon color, topic of research in numerous fields of science (e.g., physics, biology, psychology, computer vision, etc.).

In this paper, we focus on the use of colors in the field of *Content-Based Image Retrieval (CBIR)* [6, 7]. On the one hand, one has to take into account the RGB-color space used by the computer, the environmental conditions, etc. On the other hand, human color perception is of utmost importance. Since (human) users judge the retrieval results, the CBIR's matching algorithms need to provide a match that the user would accept. The complexity of this constraint is illustrated by the amount of available color spaces, such as: RGB, HSV, CIE[2] XYZ, and

[1] *Onze Taal Taalkalender 2003* (http://www.onzetaal.nl/) and
Merriam Websters Online Dictionary (http://www.m-w.com/)
[2] http://www.cie.co.at/ciecb/

Munsell[3] [3]. However, none of these color spaces model human color perception adequately.

In our opinion, one should consider color in CBIR from another perspective, that of the *focal colors* or *color categories*: Black, white, red, green, yellow, blue, brown, purple, pink, orange, and gray [2, 4, 5, 8]. People use these categories when thinking, speaking, and remembering colors. Research from diverse fields of science emphasize the importance of them in human color perception. The use of this knowledge can possibly provide a solution for the problems of color matching in CBIR.

Most CBIR-engines distinguish two forms of querying, in which the user uses either an example image (*query-by-example*) or defines features by heart, such as: shape, color, texture, and spatial characteristics (query-by-content). In the latter case, we are especially interested in *query-by-color*. At the foundation of each of these queries lies a cognitive processes, respectively color discrimination and color memory. Let us illustrate the importance of the distinction between *query-by-example* and *query-by-color* by a simple example. Imagine you want to find images of brown horses.

In the case of *query-by-example*, the resulting images will be matched on the example image: a process of color discrimination is triggered. In this process the colors are (directly) compared to each other.

In the case of *query-by-color* we need to try to imagine the color brown. Probably, you will not have a clear color in mind, but a fuzzy idea or a fuzzy set of colors: a *color category*, based on your *color memory*. Each of the elements of this brown set (or category) are acceptable colors. There is no need for several types of brown. Providing the keyword "brown" or pressing a button resembling the fuzzy set brown is sufficient.

In both forms of querying the CBIR-system can use a *Color Look-Up Table (CLUT)* for the determination of the elements of this set, described by R, G, and B-values. The set is fuzzy due to the several influences on the color (of the object of interest), such as the color of the surrounding and the semantic context in which the object is present.

However, it is clear that a distinction should be made between color categorization by discrimination and color categorization by memory. An important distinction because humans are capable of discriminating millions of colors but when asked to categorize them by memory, they use a small set colors: *focal colors* or *color categories* [2, 4, 5, 8]. Despite the fact that the importance of such a distinction is evident, this differentiation is not made in CBIR-systems.

In the remainder of this paper a question posed and two experiments executed, will be discussed. The question posed to the subjects is: "Please write down the first 10 colors that come to mind.". With the experiments we prove the difference between color categorization by color discrimination and by color memory. Hence, this research will prove that:

- The use of *color categories* is valid in a CBIR context,
- The RGB-color space can be described using *color categories*,

[3] http://www.munsell.com

- There is a difference in color categorization using color discrimination or color memory.

Moreover, we will present markers, by which the color space is divided, on which a *CLUT* for CBIR can be employed. With that a new model of human color categorization is introduced.

2 Method

2.1 Subjects

Twenty-six subjects with normal or corrected-to-normal vision and no color deficiencies, participated. They participated either voluntary or within the scope of a course. The first group were employees and the latter were students of the University of Nijmegen. They were naive as to the exact purpose of the experiment.

2.2 Equipment

An attempt was made to create an *average office environment*. Stimuli were presented on a 17" CRT monitor (ELO Touchsystems Inc., model: ET1725C), with a resolution of 1024 x 768 pixels at a refresh-rate of 75Hz. The experiment was conducted in an room with average office lighting: a Cool White Fluorescent light source: TL84 was present, its color temperature: 4100K (Narrow Band Fluorescent), as used primarily in European and Asian office lighting.

The experiments ran on a PC with an Intel Pentium II 450 MHz processor, 128mb RAM, a Matrox Millennium G200 AGP card, and with a Logitech 3-button Mouseman (model: M-S43) as pointing-device. The experiments were conducted in a browser-environment with Internet Explorer 6.0 as browser and Windows 98 Second edition as operating system, using 16-bit colors.

2.3 Stimuli

The stimuli were the full set of *the 216 web-safe colors*[4]. These are defined as follows: The R, G, and B dimensions (coordinates) are treated equally. Their minimum value is 0, the maximum value of each of the dimensions is 255. For each dimension 6 values are chosen on equal distance, starting with 0. So, for the RGB-values 0 (0%), 51 (20%), 102 (40%), 153 (60%), 204 (80%), and 255 (100%) are chosen. Each of these 6 values is combined with each of the 6 values of the 2 other dimensions. This results in $6^3 (= 216)$ triple of coordinates in the RGB-space. These RGB-values result for both Internet Explorer and Netscape under both the Windows and the Mac operating system, in the same (non-dithered) colors iff the operating system uses at least 8-bit (256) colors.

The stimulus (width 9.5 cm and height 6.0 cm) was presented in the center of the screen, on a gray background. Below the stimulus 11 buttons were placed (width: 1.8 cm and height 1.2 cm; width between: 0.6 cm). In the color memory

[4] http://www.vu.msu.edu/pearls/color/1.htm

experiment the buttons were labeled with the names of the 11 focal colors; in the color discrimination experiment each of the buttons did have one of the 11 focal colors. The 11 *focal colors* were presented conform the sRGB standard of the World Wide Web consortium (W3C)[5]. The button of choice was selected with one click of the mouse upon it.

2.4 Design

Half of the participants started with the color discrimination experiment, the other half started with the color memory experiment. Each experiment consisted of 4 blocks of repetitions of all stimuli (in a different order), preceded by a practice session. Each block consisted of the same 216 stimuli, randomized for each block and for each participant. In addition, the 11 buttons were also randomized for block and for each participant. The practice session consisted of 10 stimuli. Block, stimulus, and button order was the same for both experiments.Between the stimuli a blank screen was provided for one second, with a gray color.

The participants were asked to take a short break between the blocks of repetition, within each experiment and to take a somewhat longer break between both experiments. The duration of the breaks was determined by the subjects. In total a complete session took on the average 70 minutes, including breaks.

2.5 Procedure

The global scope of the experiment was explained, in which the experiments were conducted. After that a small questionnaire was completed. The first task was to write down the 10 colors that arise from memory first. Next, the design of the experiments was explained. The subjects were instructed for the color memory experiment to categorize the stimulus into one of the color categories, represented by their names. In the color discrimination experiment the subjects were asked to choose one of the 11 focal-colors that best resembled the stimulus. Last, was emphasized that there were no wrong answers and that if questions would arise they could be asked during one of the breaks.

3 Results

3.1 Mentioning of color names

For the determination of the confidence intervals we have used *the modified Wald method* [1] that proved to work well with a limited number experiments and with proportions close to 0 or 1.0; both the case in the present research. The proportion or frequency of appearance was determined by:

$$p = \frac{S + 2}{N + 4}$$

[5]http://www.w3.org/Graphics/Color/sRGB.html

where p is the proportion, S is the number of times the color is mentioned, and N is the number of subjects (26 in the present research).

The confidence interval was determined by:

$$p - \phi\sqrt{\frac{p\,(1-p)}{N+4}} \quad to \quad p + \phi\sqrt{\frac{p\,(1-p)}{N+4}}$$

where ϕ is 2.58 or 1.96 (in literature frequently rounded to 2.5 and 2 respectively) for the critical values from the Gaussian distribution for respectively 99% and 95%. The (relative) frequencies as well as the confidence intervals (both 99% and 95%) for all colors mentioned, are given in Table 1.

Table 1: Frequency and confidence-intervals of color names mentioned.

Color name	Frequency (in %)	min.-max. p at 99% (in %)	min.-max. p at 95% (in %)
red	26 (100.0%)	81.6% - 100.0%	84.4% - 100.0%
green	26 (100.0%)	81.6% - 100.0%	84.4% - 100.0%
yellow	26 (100.0%)	81.6% - 100.0%	84.4% - 100.0%
blue	26 (100.0%)	81.6% - 100.0%	84.4% - 100.0%
purple	24 (92.3%)	70.6% - 100.0%	74.5% - 98.8%
orange	22 (84.6%)	61.2% - 98.8%	65.7% - 94.3%
black	20 (76.9%)	52.5% - 94.1%	57.5% - 89.2%
white	20 (76.9%)	52.5% - 94.1%	57.5% - 89.2%
brown	20 (76.9%)	52.5% - 94.1%	57.5% - 89.2%
gray	15 (57.7%)	33.4% - 80.0%	38.9% - 74.4%
pink	11 (42.3%)	20.0% - 66.6%	25.6% - 61.1%
violet	06 (23.1%)	5.9% - 47.5%	10.8% - 42.5%
beige	04 (15.4%)	1.2% - 38.8%	5.7% - 34.3%
ocher	03 (11.5%)	0.9% - 34.2%	3.3% - 30.0%
turquoise	02 (7.7%)	2.7% - 29.3%	1.1% - 25.5%
magenta	02 (7.7%)	2.7% - 29.3%	1.1% - 25.5%
indigo	02 (7.7%)	2.7% - 29.3%	1.1% - 25.5%
cyan	02 (7.7%)	2.7% - 29.3%	1.1% - 25.5%
silver	01 (3.8%)	4.1% - 24.1%	0.7% - 20.7%
gold	01 (3.8%)	4.1% - 24.1%	0.7% - 20.7%
bordeaux-red	01 (3.8%)	4.1% - 24.1%	0.7% - 20.7%

There were some observations of the experimenter of possible factors of influence on the data provided by the question of mentioning 10 colors:

- Most subjects were directly able to write down 7, 8, or 9 color names, but experienced it as difficult to mention the last.
- A considerable number of participants asked whether black, gray, and white were colors during their task of writing down 10 color names. This was confirmed by the researcher who conducted the experiment.
- Another group of subjects indicated after they had written down the color names that their opinion was that these black, gray, and white are no colors. With that as opinion they had chosen to not write down black, gray, and

white. This explains for a large part the less frequently mentioned colors, most written down last.

As presented in Table 1, every subject named red, green, blue, and yellow. With 11 occurrences, pink was the least mentioned *focal color*. Nevertheless, pink was mentioned almost twice as much than the most frequently mentioned *non-focal color*: violet (6). The other *non-focal colors* were mentioned even less. In addition, the three observations mentioned above only confirms the existence of the *focal colors* in human memory.

3.2 The color discrimination experiment and the color memory experiment separate

The main result of both experiments is a table of markers for a $CLUT$[6]. The table distinguishes the discrimination and memory experiment.

We have analyzed the color discrimination experiment on each of the three dimensions: R, G, and B. Block was, on the average, a strong factor of influence on all three dimensions ($R : F(33, 192.21) = 917.90, p < .000; G : F(33, 192.21) = 1143.350, p < .000; B : F(33, 192.21) = 600.28, p < .000$). This held for all 11 color categories.

The same was done for the color memory experiment. Again block appeared a strong factor of influence on all three dimensions ($R : F(33, 192.21) = 756.54, p < .000; G : F(33, 192.21) = 785.99, p < .000; B : F(33, 192.21) = 451.35, p < .000$). Again this held for all 11 color categories.

3.3 The color discrimination and the color memory experiment together

The analysis of the experiments, conducted on the three dimensions: R, G, and B, showed a strong difference between the experiments on each of the three dimensions ($R : F(11, 15) = 2.96, p < .027; G : F(11, 15) = 7.843, p < .000; B : F(11, 15) = 3.11, p < .022$).

A more detailed analysis for each color category separate on the R dimension revealed that only purple ($F(1, 25) = 6.49, p < .017$) and red ($F(1, 25) = 20.50, p < .000$) were clearly under influence of the difference in buttons between both experiments; blue ($F(1, 25) = 3.48, p < .075$) and brown ($F(1, 25) = 3.74, p < .065$) showed only a tendency of influence.

On the G dimension all color categories, except gray and yellow, were strongly influenced by the difference in buttons between both experiments ($blue : F(1, 25) = 35.46, p < .000; brown : F(1, 25) = 33.52, p < .000; green : F(1, 25) = 21.79, p < .000; orange : F(1, 25) = 30.12, p < .000; purple : F(1, 25) = 15.91, p < .001; red : F(1, 25) = 12.58, p < .002; white : F(1, 25) = 22.26, p < .000; black : F(1, 25) = 35.27, p < .001$).

[6]The full table of markers for the $CLUT$can be found at:
http://eidetic.cogsci.kun.nl/egon/demos/vindx_colorselector/

Last, on the B dimension 6 color categories were strongly influenced by the difference between the experiments ($blue : F(1,25) = 7.67, p < .010; brown : F(1,25) = 8.67, p < .007; yellow : F(1,25) = 7,67, p < .010; pink : F(1,25) = 9.82, p < .004; white : F(1,25) = 7.19, p < .013; black : F(1,25) = 12.89, p < .001$) and orange showed a tendency of being influenced ($F(1,25) = 4.02, p < .056$); gray, green, purple and red were not influenced at all.

However, it is much more interesting to consider the colors independent of their (R, G, and B) dimensions. In both experiments (the overlap), 62 of the same web-safe colors were categorized as blue, 69 were categorized as green, and 49 were categorized as purple. For the first two of these color categories the difference in categorization between the experiments was marginal, for the latter a clear difference between both experiments was present. The remaining colors were categorized to one of the other 9 color categories. The overlap between both experiments for these categories was much smaller (average: 12.89; range: 4-20). The differences were large (average: 6.78; range: 1-19).

4 Discussion

The questionaire proved that the 11 color categories exist. This validated not only the choice of the 11 buttons used for the categorization of stimuli in the experiment, but, more importantly, it validated the idea to describe the RGB-color space using these color categories. When people use color categories when thinking, speaking, and remembering colors [2, 4, 5, 8], why not use them for describing the color space and use this description for CBIR? Since the existence of color categories proved to be valid we used them for two experiments on color categorization: one by way of color discrimination and the other by way of color memory.

Conform the hypothesis, no consistent color categorization was found *over* the experiments. This, despite the fact that the same stimuli were presented in the same blocks with the same button order, for each of the experiments. So, this leaves as conclusion that the cognitive processes of discrimination and memory influence color categorization strongly. Such a distinction argues in favor of different algorithms for color matching in CBIR, using on the one hand *query-by-example* and on the other hand *query-by-color*.

Color matching using a *CLUT*, based on the markers derived from the experimental results, could enhance the color matching process significantly. Results based on such a CLUT would be more intuitive for users. This would yield for the user more satisfying results than when using non-intuitive color matching functions founded on a color space.

Furthermore, the strong effect of the stimulus order on their perception was remarkable. This again indicates the strong influence of color memory on color perception. However, this did not explain that the *CLUT* markers define fuzzy boundaries between the color categories. This is due to a wide range of variables influencing color perception: memory, illumination, object identity, culture, emotion, and language [2, 4, 5, 8].

So, we have presented a division of the RGB-colorspace, that can be employed

as a model of human color categorization founded on two different cognitive processes: color discrimination and color memory. We propose to implement separate color matching algorithms for *query-by-example* and *query-by-color*. Each comprising their own sustained basic color categories as fuzzy clusters in the CLUT. Such an approach would yield perceptually intuitive retrieval, and with that, satisfying results for the user.

5 Acknowledgments

The Dutch organization for scientific research (*NWO*) is gratefully acknowledged for funding Eidetic (project-number: 634.000.001), a project within the ToKeN2000 research line, in which this research was done. Furthermore, we would like to thank Leon van den Broek for advice of all kinds and Frans Gremmen for advice of statistical nature.

References

[1] A. Agresti and B.A. Coull. Approximate is better than exact for interval estimation of binomial proportions. *The American Statistician*, 52:119–126, 1998.

[2] B. Berlin and P. Kay. *Basic color terms: Their universals and evolution*. Berkeley: University of California Press, 1969.

[3] S. Douglas and T. Kirkpatrick. Do color models really make a difference? In M. J. Tauber, V. Bellotti, R. Jeffries, J. D. Mackinlay, and J. Nielsen, editors, *Proceedings of CHI 96: Conference on Human Factors in Computing Systems*, pages 399–405. ACM, New York: ACM Press, 1996.

[4] R.L. Goldstone. Effects of categorization on color perception. *Psychological Science*, 5(6):298–304, 1995.

[5] E. Rosch Heider. Universals in color naming and memory. *Journal of Experimental Psychology*, 93(1):10–20, 1972.

[6] Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Past, present, and future. *Journal of Visual Communication and Image Representation*, 10:1–23, 1999.

[7] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[8] E.L. van den Broek, L. G. Vuurpijl, P. Kisters, and J. C. M. von Schmid. Content-based image retrieval: Color-selection exploited. In M.-F. Moens, R. de Busser, D. Hiemstra, and W. Kraaij, editors, *Proceedings of the Dutch-Belgian Information Retrieval Workshop (DIR2002)*, volume 3, pages 37–46, Belgium, Leuven, December 2002. Catholic University Leuven, Belgium.

Application of Inductive Concept Learning to Doctor-Patient Relation Data

Raquel Costa [a] Federico Divina [b]

[a] GGZ Mentrum, Amsterdam
raquel.costa@mentrum.nl
[b] Dept. Computer Science, Vrije Universiteit, Amsterdam
divina@cs.vu.nl

Abstract

This paper proposes a preliminary study of a new, real life propositional dataset, regarding the relation between patients and their doctors. The aim of this paper is to analyze this dataset from two point of view: the satisfaction of the patient with his or her relation with the doctor and the problem of recognizing whether a patient is admitted in a psychiatric ward or not. The first problem is an important one since the measure of the quality in the health care providing is gaining more and more attention. The second problem allows to check if psychiatric patients differ from other patients in social context, how they are treated by doctors and what they want from their treatment. The Inductive Logic Programming algorithm ECL is employed for inducing rules for the two addressed problems. We have chosen to use ECL because its outputs is of easy comprehension for experts in the psychiatric field.

1 Introduction

In the health care providing, satisfaction is earning more and more importance. Patients are nowadays better informed about their rights and the advances in medicine. They ask more from the doctor; the time when the patient believed everything that the doctor said is over. The traditional doctor-patient relationship is becoming a client-provider relationship. Patients go to the doctor for advice and then decide about their treatment. They analyze critically what the doctor says. At the same time, they know that they may choose between different care providers. The way in which doctors approach patients becomes this way an important factor. The client is king. Because of this we have decided to collect opinions from different patients with an interview questionnaire. The objective was to measure what the features of the doctor-patient relationship in different wards are and how different patients perceive it. Also the degree of satisfaction among the patients and the determinant factors for satisfaction were included in our objective. In our study we have taken two kinds of patients: psychiatric patients and non-psychiatric patients (belonging to the internal medicine and general surgery wards). The reason for these two groups is the special character of the

doctor-patient relationship in the psychiatry. All the collected information has been organized in a dataset [2]. In every branch of the medicine, the relation between doctor and patient is of great importance. It determines the compliance of the treatment and a part of the curative process. In the psychiatry the therapeutic relationship has even more power. Therefore having a general rule that could guide doctors towards a good relation with their patients would be very useful. The communication between doctor and patient would be better and that would lead to more satisfaction, better compliance with the treatments and better prevention [1]. That means an optimal use of health care services. Such a rule could be induced with Inductive Concept Learning (ICL) tools. ICL [9] constitutes a central topic in Machine Learning. The problem can be formulated in the following manner: given a description language used to express possible hypotheses, a background knowledge, a set of positive examples, and a set of negative examples, one has to find a hypothesis which covers all positive examples and none of the negative ones (cf. [7, 10]). The so learned concept can be used to classify previously unseen examples. Concepts are induced because obtained from the observation of a limited set of training examples. When hypotheses are expressed in (a fragment of) first order logic, ICL is called Inductive Logic Programming (ILP).

In this paper we try to induce a satisfaction rule from the data with a ILP system: ECL [3, 4]. Moreover we analyze the data also to check whether psychiatric patients differ from other patients in their answers to the questionnaire. We do that by trying to induce a rule that can distinguish between psychiatric patients and non-psychiatric patients. If such a rule exists, then it can identifies the aspects of the relation doctor-patient that are different when the patient is a psychiatric one.

This paper is organized as follow. In section 2 we describe the dataset and the two problems tackled in this paper. In section 3 we present the method we use for analyzing the data and in section 4 we present and discuss the obtained results. Finally in section 5 we give some conclusions.

2 The Dataset

The data was collected between November 2000 and June 2001 in the psychiatry ward of Hospital Gil Casares, Santiago de Compostela, Spain and in the internal medicine and general surgery wards of the Hospital Xeral, Vigo, Spain. The same interviewer collected all data. Ninety patients were interviewed in the hospitals following an interview questionnaire. The interview questionnaire was specially designed for this research project. The reason is that this questionnaire measures the satisfaction only from the doctor-patient relation point of view. Other questionnaires published in the medical literature, e.g the Verona service satisfaction scale (VSSS) [11], take more elements into account. Our questionnaire is based on the VSSS, with less items to facilitate the recruitment of admitted patients and focusing on the doctor-patient relationship as a satisfaction determinant factor. The questionnaire consists on ten variables referred to patient's demographic data and twenty-six items that feature the relation between the doctor and the patient.

It was possible to answer to these twenty-six items with five different grades.

Thus each patient is described by thirty-six attributes. Moreover it is also known in which ward the patient was admitted. The ten demographic attributes include the sex, the age, the marital status (single, married...), the number of children, the place of residence, the education level, the working situation (if the patient is working or not, if the patient is retired...), the kind of profession and in which environment the patient lives (alone, with parental or own family). All this data, but the age of the patient, are discrete. The age of the patient is considered a numerical attribute.

The other twenty-six attributes include, among the others, the satisfaction level of the relation between the interviewed patient and the doctor, the way the doctor gives information to the patient (the comprehensibility of the language that the doctor uses with the patient), the frequency of the contact between doctor and patient, the degree of personal involvement of the doctor and so on. All this attributes are nominal. A detailed description of all the attributes can be found in [2]. For privacy reasons each patient is identified by an unique code. So examples have the following form: $case(id_1), case(id_2) \ldots$, where id_1 and id_2 are different patients. The background knowledge contains facts of the following form: *sex(id_1,m), age(id_1,30), satisfied(id_1,y), trust(id_1,1)* ..., saying that the patient identified by id_1 is a male of thirty years, satisfied with his relation with his doctor and that he considers that he can trust his doctor with a degree equal to 1, and so on. We have chosen to address two problems:

Satisfaction Problem Patient satisfaction is related to the quality of the patient care. It is although not the only parameter to consider in the measure of quality care [12]. Satisfaction with health care is a multidimensional concept. It includes availability, financial aspects, convenience, staff's professional skills, among others. One of the determinant factors is the relation between doctor and patient [8].

In this problem positive examples are patients that are absolutely satisfied with their relation with their doctors, and negative examples are those patients who are not completely satisfied. For this problem there are 61 positive examples and 29 negative examples. The background knowledge contains 3274 facts.

Service Problem In this problem we want to induce rules to distinguish psychiatric patients from non-psychiatric patients. Psychiatric patients may differ from other kind of patients in demographic features. Disorders as schizophrenia, alcohol and drugs abuse are more frequent in the population with a lower social status. Bipolar disorder type I is more frequent among people with lower educational level. Dementia is more frequent among the elderly [6]. Maybe these differences go further than the demographic data. Thus it would be interesting to know if psychiatric patients perceive the doctor-patient relation differently than other people.

For this problem there are 43 positive examples and 47 negative examples. The background knowledge is made of 3198 facts. There are less facts in

the background knowledge because some attributes were excluded from this problem, e.g. the diagnosis for the disease of the patient was excluded, for obvious reasons.

3 ECL: a GA based Inductive Concept Learner

The two problems are analyzed with the Inductive Logic Programming system ECL. ECL was profitably applied [4] also to propositional data, and so it is suitable for this task. Moreover the output produced by ECL (a logic program) is of easy comprehension, so that experts in the field can asses the goodness of the induced rules. ECL is a GA based inductive concept learner. This system evolves a set of Horn clauses that form a Prolog program. In figure 1 a scheme of ECL is given.

```
ALGORITHM ECL
Sel = positive_examples
repeat
    Select partial Background Knowledge
    Population = ∅
    while (not terminate) do
        Adjust examples weights
        Select n chromosomes using Sel
        for each selected chromosome chrm
            Mutate chrm
            Optimize chrm
            Insert chrm in Population
        end for
    end while
    Store Population in Final_Population
    Sel = Sel - { positive examples
        covered by clauses in Population }
until max_iter is reached
Extract final theory from Final_Population
```

Figure 1: The overall learning algorithm ECL

The system takes as input a background knowledge (BK), and a set of positive and negative examples, and outputs a set of Horn clauses that covers many positive examples and few negative ones.

Recall that a Horn clause is of the form $p(X,Y) : -r(X,Z), q(Y,a).$ with head $p(X,Y)$ and body $r(X,Z), q(Y,a)$. A clause has a declarative interpretation: $\forall X, Y, Z(r(X,Z), q(X,a) \rightarrow p(X,Y))$ and a procedural one: *in order to solve* $p(X,Y)$ *solve* $r(X,Z)$ *and* $q(Y,a)$. Thus a set of clauses forms a logic program, which can directly (in a slightly different syntax) be executed in the programming language Prolog. The background knowledge used by ECL contains ground facts (i.e. clauses of the form $r(a,b) \leftarrow .$ with a, b constants). The training set contains

facts which are true (positive examples) and false (negative examples) for the target predicate. A clause is said to *cover an example* if the theory formed by the clause and the background knowledge logically entails the example.

In the repeat statement of the algorithm a `Final_population` is iteratively built from the empty one. Each iteration performs the following actions: part of the background knowledge is randomly selected, an evolutionary algorithm that uses that part of BK is run and the resulting set of Horn clauses is joined to the actual `Final_population`. The evolutionary algorithm evolves a `Population` of Horn clauses starting from an empty population, where an individual represents a clause, by the repeated application of selection, mutation (the system does not use any crossover operator) and optimization in the following way. At each generation n individuals are selected using a variant of the US selection operator [5]. Roughly, the selection operator selects a positive example and performs a roulette wheel on the set of individuals in the `Population` that cover that example. If that example is not covered by any individual then a new clause is created using that example as seed. Each selected individual undergoes mutation and optimization. Mutation consists of the application of one of the following four generalization/specialization operators. A clause is generalized either by deleting an atom from its body or by turning a constant into a variable, and it is specialized by either adding an atom or turning a variable into a constant. Each operator has a degree of greediness. In order to make a mutation, a number of mutation possibilities is considered, and the one yielding the best improvement, in terms of fitness, is applied. Optimization consists of the repeated application of one of the mutation operators, until the fitness of the individual increases, or a maximum number of optimization steps has been reached. Individuals are then inserted in the population. If the population is not full then the individuals are simply inserted. If the population has reached its maximum size, then n tournaments are made among the individuals in the population and the resulting n worst individuals are substituted by the new individuals.

The fitness of an individual x is given by the inverse of its accuracy:

$$fitness(x) = \frac{1}{Acc(x)} = \frac{P+N}{p_x+(N-n_x)}$$

In the above formula P and N are respectively the total number of positive and negative examples, while p_x and n_x are the number of positive and negative examples covered by the individual x. We take the inverse of the accuracy, because ECL was originally designed to minimize a fitness function. When the `Final_population` is large enough (after `max_iter` iterations) a subset of its clauses is extracted in such a way that it covers as many positive examples as possible, and as few negative ones as possible.

4 Results and Discussion

In this section we will present some results obtained with ECL. In all the experiments performed with ECL all the clauses considered are allowed to have at most three atoms in their body. This limitation is necessary for allowing an easy interpretation of the clauses by experts. We first run ECL on the complete set of

examples for both problems, in order to induce useful rules according to the judgment of experts in the field. We then validate the method with a cross-validation evaluation. An argument written as $\{a, b\}$ means that it can be either a or b.

4.1 Satisfaction Problem

On the Satisfaction problem, the best logic program induced by ECL consists of seven clauses. This logic program has an accuracy of 98.89%, a precision of 98.39% and a recall of 100%. This solution was found with a population size of 50, 10 generations and 10 individuals selected per generation and using 0.2 of the background knowledge at each iteration of the GA, and max_iter set to 10.

An example of clause found is $case(X) : -wlclarity(X, \{n, cn\}), explained(X, s).$ which cover 23 positive examples and 1 negative. This clause states that if a doctor has given information about the disease to the patient and if the patient finds it comprehensible enough, then the patient is satisfied. As expected, good communication between doctor and patient determines the satisfaction of the patient. If patients think that the doctor speaks clearly enough and they perceive that the doctor explains to them their illness, they are satisfied.

An interesting clause, not found in the best solution is: *case(X):- trust(X,{cs,s}), wait(X,{n,av}).* that states the if the patient trusts his doctor and does not have to wait then is satisfied by his doctor-patient relation. This clause covers 52 positive examples and 10 negatives. As expected, the system found correlation between the patient's trust in the doctor and not having to wait and perceived satisfaction.

Another instance is: $case(X) : -change(X, n), trust(X, s), wlsd(X, \{cs, s\})$. Also this clause was expected. It says that if a patient does not want to change doctor, if he trusts his doctor and would like to be seen always by the same doctor, then the patient is satisfied. The clause covers 33 positive examples and 2 negatives.

4.2 Service Problem

On the Service problem, the best logic program induced by ECL is made of nine clauses. This logic program has an accuracy of 89%, a precision of 82.35% and a recall of 97.68%. This solution was found with a population size of 100, 10 generations, 20 individuals selected per generation, max_iter set to 5 and with half of the background knowledge used at each iteration of the GA. From a practical point of view, not all of the clauses were useful, however a number of them turned out to be correct. For example the clause $case(X) : -trust(X, n).$ which states that if a patient does not trust his or her doctor then the patient is more likely to be a psychiatric patient. This clause correctly covers 9 positive examples and no negative ones. The clause is judged to be correct. In fact psychiatric patients are sometimes compulsory hospitalized. Because they have not chosen to see that doctor, it can be justified that they do not trust him. Moreover, some psychiatric patients suffer from delusions (false beliefs which are held with perseveration). One typical delusion is the belief that everyone in the environment is an enemy.

That would explain that the patient sees the doctor as an enemy and does not trust him.

Another interesting clause is $case(X) : -explained(X, n), same\_doc(X, \{cs, s\})$. that states that if the patient affirm that his doctor has never explained him his disease and if the patient is visited almost always by the same doctor, then it is likely to be a psychiatric patient. This clause covered 14 positive examples and 1 negative. Patients with a psychiatric disorder have frequently lack of insight. They do not perceive properly what their illness is. That can explain why they answer more than other patients that nobody has explained them what their illness is. If they are always seen by the same doctor, that is even more likely. It could be that patients do not get an explanation of their illness in the first contact , but it should certainly happen after a few meetings.

The last example we propose for the Service problem is represented by the clause $case(X) : -work(X, \{baja, pensionista\})$.. This clause covered 6 positive examples and no negative. It describes the working situation of the patient, saying that if a patient has a temporal or permanent situation of work inability or he is retired, then he is likely to be a psychiatric patient. Patients with a psychiatric disorder are impaired in different functional areas, one of them is work. That justifies the association between temporal or permanent situation of work inability and retiring, and psychiatric disorder.

4.3 Cross-validation

We have evaluated ECL on the problems with a ten-fold cross-validation method. Each dataset is divided in ten disjoint sets of similar size; one of these sets is used as a test set, and the union of the remaining nine forms the training set. ECL is then run, with the same parameter used in the previous section, on the training set and the resulting logic program, is tested on new examples using the test set. Three runs with different random seeds are performed on each dataset.

Problem	ECL	C4.5
Satisfaction	0.72 (0.08)	0.68 (0.04)
Service	0.70 (0.03)	0.72 (0.02)

Table 1: Results for the two problems. For both problem the average accuracy and standard deviation (between brackets) are shown.

The average accuracy and standard deviation (between brackets), over all three runs, are given in table 1. The same table present also the same information on the results obtained by C4.5. No limitation on the number of attributes that can be taken into consideration by C4.5 was used. This can explain the difference in the performance of the two systems on the Service problem. However even in this case the accuracy of the solution found by ECL is comparable with the one found by C4.5. The computational cost is the main drawback of ECL, in fact C4.5 is much faster than ECL.

5 Conclusions

In this paper we have analyzed a new real life dataset regarding the relation doctor-patient. We have derived two problems from this dataset, the satisfaction of the patient with his or her relation with the doctor and the problem of identifying psychiatric patients. This last problem is important because analyzing the induced rules, experts can check what the differences are in the doctor-patient relation with psychiatric patients. Rules obtained on the satisfaction problem are the expected ones. They focus on concepts like the will of the patient to change doctor, or the will to be seen by the same doctor. All this factors are the expected ones. For the service problem, results suggests that psychiatric patients may perceive the relation with their doctor in a different way than other kind of patients.

In this paper we have proposed a preliminary analysis of the data, using only ECL. We have also applied C4.5 but its results were only used for comparison in the cross-validation. As a future work we are planning to analyze the data with other classifier systems.

References

[1] D. A. Barker, S. S. Shergill, I. Higginson, and M. W. Orrel. Patients' views towards care received from psychiatrists. *British Journal of Psychiatry*, 168:641–646, 1996.

[2] R. Costa and F. Divina. Doctor-patient relation (www.cs.vu.nl/∼divina/), 2003.

[3] F. Divina and E. Marchiori. Evolutionary concept learning. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 343–350, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[4] Federico Divina, Maarten Keijzer, and Elena Marchiori. A method for handling numerical attributes in GA-based inductive concept learners. In *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, pages 898–908, Chicago, 12-16 July 2003. Springer-Verlag.

[5] A. Giordana and F. Neri. Search-intensive concept induction. *Evolutionary Computation*, 3(4):375–416, 1996.

[6] H. I. Kaplan, B. J. Sadock, and J. A. Grebb. *Synopsis of Psychiatry*. Williams and Wilkins, 1994.

[7] M. Kubat, I. Bratko, and R.S. Michalski. A review of Machine Learning Methods. In R.S. Michalski, I. Bratko, and M. Kubat, editors, *Machine Learning and Data Mining*. John Wiley and Sons Ltd., Chichester, 1998.

[8] J. J. Mira. La satisfaccion del paciente, 2001.

[9] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[10] T.M. Mitchell. *Machine Learning*. Series in Computer Science. McGraw-Hill, 1997.

[11] M. Ruggeri and R. Dall'Agnola. Verona service satisfaction scale (vsss-54), 1993.

[12] S. R. Weingarten, E. Stone, A. Green, M. Pelter, S. Nessim, H. Huang, and R. Kristopaitis. A study of patient satisfaction and adherence to preventive care practice guidelines. *The American Journal of Medicine*, 99:590–595, 1995.

Reactive agents and perceptual ambiguity

Michel van Dartel, Ida Sprinkhuizen-Kuyper,
Eric Postma, and Jaap van den Herik

IKAT, Universiteit Maastricht[1]

Abstract

Situated and embodied reactive agents solve relatively complex tasks by coordinating action and perception. Reactive agents are generally believed to be incapable of coping with identical sensory states that require different responses (i.e., perceptual ambiguity). In contrast to reactive agents, non-reactive agents can cope with perceptual ambiguity by storing and integrating sensory information over time. This paper investigates how and to what extent reactive agents can cope with perceptual ambiguity. An active categorical perception model is introduced in which agents categorise objects by coordinating action and perception. The agent's neurocontrollers are evolutionary optimised for the task. Our results show that reactive agents can cope with perceptual ambiguity despite their incapability to store sensory information over time. An analysis of behaviour reveals that reactive agents use the environment as an external memory.

1 Introduction

Perceptual ambiguity occurs when identical sensory states require different responses. Ambiguous sensory states cause a problem for agents that have to select an appropriate action. The problem of perceptual ambiguity can be solved by storing information over time [1]. In agent models of behaviour, reactive agents, i.e., agents reacting directly and exclusively to the current sensory state, are generally regarded as unable to cope with perceptual ambiguity. Beer [1] claims that only non-reactive agents, i.e., agents with internal (recursive) dynamics, can cope with perceptually ambiguous tasks. He states that they can do so because their internal dynamics allow non-reactive agents to organise 'their behaviour according to sensory stimuli that are no longer present' (p. 424). However, numerous animals behave reactively [6] and are still able to cope with perceptual ambiguity, presumably omnipresent in their natural environments. The research question addressed in this paper reads: How and to what extent can reactive and non-reactive agents cope with perceptual ambiguity?

To answer this question, we define a model of an agent embedded in a simple environment in which the level of perceptual ambiguity can be varied. The sensor configuration of the agent determines the level of perceptual ambiguity. The model is called the active categorical perception model (cf. [2]) to emphasise the

[1]P.O.Box 616, 6200 MD Maastricht, The Netherlands. E-mail: {mf.vandartel, kuyper, postma, herik}@cs.unimaas.nl

active nature of perception in the agents studied. Through simulation studies, the ability of reactive agents to cope with different levels of perceptual ambiguity is evaluated.
The outline of this paper is as follows. In section 2, the active categorical perception model is presented. Section 3 describes the experiments conducted and reports on the results. In section 4, the behaviour of optimised agents is analysed. The results are discussed in terms of internal and external memory and conclusions are provided in section 5.

2 The active categorical perception model

In the active categorical perception model, agents are optimised to catch and avoid falling objects. In the model, categorical perception is active because it is achieved through an active interaction between agent and environment. Active perception requires that the model is situated, i.e., that the agent is able to interact with its environment.
A two-dimensional grid G_t of size $x_{max} \times y_{max}$ defines the environment in which the agent acts at time t. For all experiments we set $x_{max} = 20$ and $y_{max} = 10$. The objects and agents are allowed to move through the left and right boundaries of the environment, defined by $x = 0$ and $x = x_{max} - 1$, and to re-appear at the opposite side of the environment. In the model, time passes in discrete steps. Each object is placed at $y = y_{max} - 1$ at $t = 0$. The law of gravity in the model causes objects to fall. Objects hit the floor defined by $y = 0$ at $t = y_{max} - 1$. An object is represented by a sequence of ones: $G_t((x + j)\ mod\ x_{max}, y) = 1$ for $j \in \{0, 1, \ldots j_{max} - 1\}$, with j_{max} the width of the object. Two types of objects are defined: small objects ($j_{max} = 2$) and large objects ($j_{max} = 4$). For the initial horizontal position of an object, x is selected from $x \in \{0, 1, \ldots x_{max} - 1\}$. The dynamics of falling objects are represented as follows. From t to $t + 1$ grid G_t is modified by the following rules: $G_{t+1}(x, y) = G_t((x + 2d)\ mod\ x_{max}, y - 1) \quad (x = 0, 1, \ldots x_{max} - 1)$, and $G_{t+1}(x, y - 1) = 0 \quad (x = 0, 1, \ldots x_{max} - 1)$, with $d \in \{-1, +1\}$ a direction parameter that is fixed between $t = 0$ and $t = t_{max}$. The first formula defines the new position of the object. The second erases the old position of the object by setting all values of grid cells in the row of grid G_t in which the object was last positioned to zero. The object defined by the sequence of ones in G_t moves leftward for $d = -1$ and rightward for $d = +1$. Figure 1 illustrates four consecutive simulation time steps (denoted by t to $t + 3$) in the active categorical perception model with a large object (represented by black grid cells) falling leftward. The four circles in the bottom row of each grid represent the sensors of the agent, which are activated (grey circles) by the presence of an object in the same column. In the figure, the agent moves four grid cells leftward in each time-step.
An agent consists of a neurocontroller that receives environmental input through an array of sensors. A motor system moves the agent according to the output of the neurocontroller. Two feedforward neural networks served as neurocontrollers for the reactive agent: (1) a simple perceptron (P) and (2) a simple multilayer perceptron (MLP) of equal size.

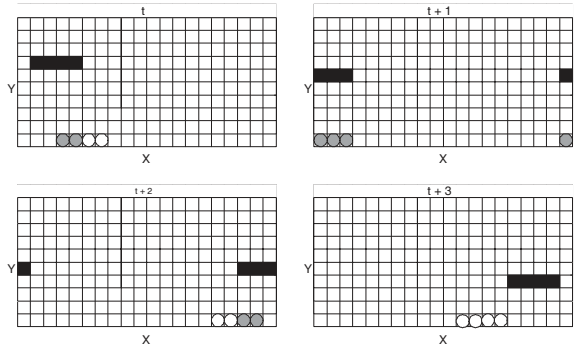


Figure 1: An agent with inactive sensors (white circles) and activated sensors (grey circles) as a result of detecting a large object (black grid cells) in the environment (grid G) over four time steps.

The sensor array of the agent consists of two types of sensors: functional sensors and blind sensors. Each agent has s functional sensors ($s = 4$, for all experiments outlined in section 3), with each functional sensor connected to one input node. A variable number of blind sensors b ($b \in \{0, 1, 2, 3\}$), i.e., sensors that are not connected to the input layer, are inserted into the sensor array; they determine the level of perceptual ambiguity. Blind sensors are always positioned in the middle of the sensor array. The sensors occupy neighbouring grid cells and are constrained to the bottom row of the grid G_t, $G_t(x, 0)$ for all t. The activation of the sensor at position x at time t is represented by $I(x, t)$ and is defined as: $I(x, t) = \sum_{y=0}^{y_{max}-1} G_t(x, y)$. In order to catch and avoid objects, agents can move to the left, move to the right, or stand still in the bottom row of G_t. Since the environment is circular, i.e., objects and agents can move through the walls and re-appear at the opposite side of the environment, the output can have any value. The output of the neurocontroller, rounded to the nearest integer value, defines the number of steps st moved to the left (negative output) or right (positive output). If $st = 0$, the agent does not move. For a sensor positioned at x, the new position after movement is defined by: $(x + st)\ mod\ x_{max}$. The behaviour of an agent is evaluated when the object reaches the bottom row of grid G_t at $t = y_{max} - 1$. An object is caught by the agent *iff* $|ca - co| \leq 4.5$ (modulo the boundaries), with ca representing the centre of the agent and co the centre of the object; an object is avoided *iff* $|ca - co| > 4.5$ (modulo the boundaries). Setting the criteria for behavioural evaluation in this way ensures that objects that activate sensors at $t = y_{max} - 1$ are evaluated as being caught. The agent's task is to categorise the two classes of objects (small and large). Agents are optimised to avoid large objects and to catch small objects. By avoiding large objects and catching small ones an agent exhibits its ability to categorise.

Perceptual ambiguity is defined as the proportion of sensory states that are ambiguous to the agent. Sensory states are ambiguous when they can be caused by

the presence of an object from either object class (small or large). Hence, categorisation is not possible on the basis of a single ambiguous sensory state. For instance, in figure 1 all depicted sensory states are ambiguous to the agent except for the sensory state at time $t+1$, since this state can only result from the presence of a large object.
The number of blind sensors inserted in the middle of the sensor array determines the level of perceptual ambiguity. When the number of blind sensors is increased, the proportion of ambiguous sensory states increases too, i.e., it becomes more difficult for the agent to categorise objects. Table 1 shows the percentages of unique and ambiguous sensory states for the different numbers of blind sensors ($b \in \{0, 1, 2, 3\}$). Only nine or fewer different sensory states (depending on the number of blind sensors, as shown in table 1) can occur in the agent's sensor array during the task. The percentages listed in table 1 were obtained by recording the

number of blind sensors	number of possible sensory states	number of unique sensory states	number of ambiguous sensory states
$b = 0$	9	4 (44.4%)	5 (55.6%)
$b = 1$	9	4 (44.4%)	5 (55.6%)
$b = 2$	8	3 (37.5%)	5 (62.5%)
$b = 3$	7	0 (0%)	7 (100%)

Table 1: Number of possible, unique, and ambiguous sensory states for $b \in \{0, 1, 2, 3\}$ blind sensors.

occurrences of each sensory state while exhaustively generating all possible sensory states of objects from both classes.
The behaviour of the reactive agents depends on the weight values of their constituent neurocontrollers. We employ an evolutionary algorithm to optimise the behaviour of agents in our model. An evolutionary algorithm determines the weight values for all connections in the agent's neurocontroller. After randomly initialising the weights of the neurocontrollers, the complete generation is tested on the active categorical perception task described above. An agent's fitness F, i.e., the success of a tested agent, is calculated as: $F = (CC + CA) - (FC + FA)$, with CC the sum of correctly caught objects, CA the sum of correctly avoided objects, FC the sum of caught objects that should have been avoided, and FA the sum of avoided objects that should have been caught. Agents are tested on all possible starting positions, object classes, and directions in which objects can fall (i.e., $x_{max} \cdot 2 \cdot 2 = 80$ trials). The performance of an agent is expressed by its success rate ($\in [0, 1]$), which is calculated by $(F + 80)/(2 \cdot 80)$. The algorithm uses the standard evolutionary techniques of reproduction, crossover, and mutation [3]. By constraining the input weights of all neurocontrollers we reduced the evolutionary search space. For all agents the value of the i-th input weight to the left of the centre is defined to be equal to minus the value of the i-th input weight to the right of the centre ($i \in$ 1,2).

3 Experiments and results

Experiments were performed with each combination of neuro-controller and number of blind sensors. Each experiment was executed 5 times, over which the success rates of the best-performing agents were averaged. Comparing the results of the 8 experiments provides insight into the ability to cope with perceptual ambiguity. Table 2 shows the average success rate ($\overline{sr}$) and standard deviation (sd) for all combinations of neurocontroller and number of blind sensors (b). All agents appear

	b=0		b=1		b=2		b=3	
	$\overline{sr}$	sd	$\overline{sr}$	sd	$\overline{sr}$	sd	$\overline{sr}$	sd
P	0.7500	0.0049	0.7875	0.0056	0.7250	0.0046	0.6100	0.0017
MLP	0.8350	0.0160	0.8375	0.0199	0.7725	0.0132	0.7225	0.0096

Table 2: Average success rate ($\overline{sr}$) and standard deviation (sd) for all neurocontrollers and sensor configurations.

capable of performing categorical perception above chance level (i.e., $\overline{sr} > 0.50$). For reactive P-controlled agents, those with one blind sensor ($b = 1$) outperform agents without blind sensors ($b = 0$). It should be noted that for both cases, the percentage of ambiguous sensory states is identical (i.e., 44.4%; see table 1). For larger numbers of blind sensors ($b = 2$ and $b = 3$) a decrease in performance is observed. The reactive MLP-controlled agents outperform the P-controlled ones for each value of b, but show a similar pattern of relative performance for different values of b.

Performance on the active categorical perception task is not linearly related to the perceptual ambiguity in the task. This is, for instance, shown by the differences in performance between agents with $b = 0$ and $b = 1$ for both neurocontrollers, while the number of ambiguous sensory states is equal for both values of b. This non-linear relation can be attributed to two factors. First, it can be attributed to the change in sensor configuration as a result of altering the number of blind sensors. Second, it can be attributed to the difference in the spatial extent or scope of the sensor array. Presumably, both factors influence the performances in our study.

4 Analysis of behaviour

We have now established to what extent reactive agents can cope with perceptual ambiguity. In this section we investigate the different strategies employed. State-transition diagrams were constructed from the sensory states of optimised agents in order to investigate the behavioural strategies of reactive agents enabling them to cope with perceptual ambiguity. Sensory state-transition (SST) diagrams make explicit how reactive agents make use of their environment.

Before SST diagrams can be constructed, sensory state-action (SSA) mappings of optimised agents have to be extracted, see figure 2. On the basis of the SSA

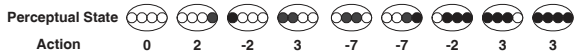
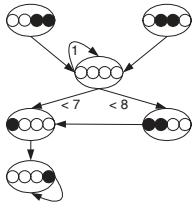
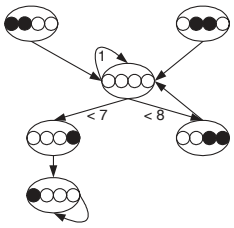


Figure 2: Mapping of sensory state to action of an optimised MLP-controlled agent without blind sensors ($b = 0$, $\overline{sr} = 0.835$).

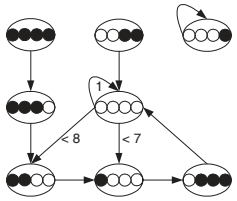
mapping, and the type and direction of movement of objects, we calculated every possible sequence of SST. In figure 3, the SST diagrams for an optimised MLP-controlled agents are shown. In the SST diagrams, and the SSA mappings used



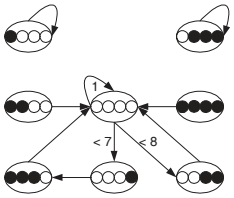
(a) Sensory state-transition diagram of an MLP-controlled agent facing a small object falling rightward.



(b) Sensory state-transition diagram of an MLP-controlled agent facing a small object falling leftward.



(c) Sensory state-transition diagram of an MLP-controlled agent facing a large object falling rightward.



(d) Sensory state-transition diagram of an MLP-controlled agent facing a large object falling leftward.

Figure 3: Sensory state-transition diagram of an optimised MLP-controlled agent and a small object falling rightward (a) or leftward (b), and with a large object falling rightward (c) or leftward (d).

to construct the diagrams, sensory states are represented by ellipses with black or white circles, expressing the binary value of activation for each individual sensor (i.e., black = '1' and white = '0'). The arrows in the diagrams represent transitions from one sensory state to another. Since the depicted agents are reactive, there can only be one successor state from each sensory state. However, this is not the case for the sensory state in which all sensors are inactive, i.e., the sensory state 0000. Although the agent behaves reactively, in the 0000 sensory state the object can be at many different positions relative to the agent. The arrow from sensory state 0000 to a target sensory state is labelled with a number indicating the number of time steps before the transition to the target sensory state occurs,

i.e., the number of time steps the agent remains in sensory state 0000.
For small objects, only 6 of the 9 sensory states depicted in figure 2 can occur. Hence, the diagrams in figures 3a and 3b, contain 6 sensory states. For large objects, 8 of them can occur, which is why the corresponding diagrams 3c, and 3d) contain 8 sensory states. Optimised MLP-controlled agents reveal behavioural attractors in the MLP-controlled agent. Besides n-cycles, their diagrams reveal the existence of point attractors that lead to catch behaviour. The point attractors correspond to sensory states 0001 and 1000 for right and leftward falling small objects (figures 3a and 3b), respectively. These attractor states also occur in the case of large objects that should be avoided (see figures 3c and 3d). The agent in figure 3 is able to enter these attractor states in almost all small-object cases (see figure 3a and 3b). However, in the large object cases, the agent only occasionally enters these attractor states (see figure 3c and 3d), namely, when the attractor state is the initial state (i.e., when $t = 0$). The behavioural attractors in figure 3 clearly demonstrate how a reactive agent can make a categorical decision on the basis of information stored in the environmental dynamics. By evolution it has learned that different classes of objects yield different consequences of actions, e.g., entering or not entering a behavioural attractor.
Our analysis reveals that reactive agents compensate their lack of internal memory by using the environment (i.e. the falling object) as an external memory. The external memory is most strickingly revealed by the attractor states that couple the dynamics of the agent to the dynamics of the object. MLP-controlled agents make use of external memory as expressed by the behavioural attractors in the SST diagrams.

5 Discussion and conlusion

Our findings contradict Beer's claim that reactive agents (i.e., agents without internal dynamics) 'cannot organise their behaviour according to sensory stimuli that are no longer present' [1] (p.224). Both reactive controllers (P and MLP) perform categorisation above chance-level, even when all sensory states are ambiguous, i.e., when $b = 3$. Nolfi (in [5]) supports the conclusion that reactive agents can perform relatively well on tasks with perceptual ambiguity. The better performance of MLP-controlled agents over the P-controlled agents indicates the importance of a non-linear sensory state-action mappings in reactive agents coping with perceptual ambiguity.
It is important to note that evolutionary selection occurs on the basis of complete sequences of sensory-motor behaviour, rather than on single perception-action steps. Consequently, our results can be explained by the fact that perceptual ambiguity is defined locally in time, while behavioural success (catching or avoiding) is defined globally.
A similar conclusion was drawn by Nolfi [5], who found reactive robots to cope with perceptual ambiguity in a task in which all sensory states were ambiguous. He concluded that they did so by exploiting the 'emergent behaviours resulting from a sequence of sensory-motor loops and the interaction between robot and the

environment' (p.119). In Nolfi's experiment [5], reactive agents were evolutionary optimised to find a goal position in a static environment. The results of our experiments extend Nolfi's results to a categorical perception task in a dynamic environment. Categorisation is often argued to be fundamental to cognition [2, 4]. In our experiments, the environment acts as an external memory for the reactive agents. Past actions are (to a certain extent) reflected in the current sensory state. Trajectories towards behavioural attractors are expressions of the external memory. Hence, in this task an internal memory is not needed to cope with perceptual ambiguity. Natural cognitive systems have also been shown to use the environment as an external memory [7].
Our results illustrate how and to what extent situated reactive agents can cope with perceptual ambiguity. The extent to which reactive agents can cope with perceptual ambiguity depends on the ability to map sensory states to actions in a non-linear way. We conclude that reactive agents can cope with perceptual ambiguity by exploiting the environment as an external memory store.

References

[1] R.D. Beer. Toward the evolution of dynamical neural networks for minimally cognitive behavior. *From Animals to Animats 4. Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, 4:421–429, 1996.

[2] R.D. Beer. The dynamics of active categorical perception in an evolved model agent. *To appear in Adaptive Behavior*, In press.

[3] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading MA: Addison-Wesley, 1989.

[4] S. Harnad. *Categorical perception: The groundwork of cognition.* New York NY: Cambridge University Press, 1987.

[5] S. Nolfi. Power and limits of reactive agents. *Neurocomputing*, 49:119–145, 2002.

[6] S. Nolfi and D. Marocco. Active perception: A sensorimotor account of object categorization. *From Animals to Animats 7. Proceedings of the VII International Conference on Simulation of Adaptive Behavior*, 2001.

[7] J.K. O'Regan and A. Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, 24(5):883–917, 2001.

A Programming Language for Cognitive Agents Goal Directed 3APL

Mehdi Dastani Birna van Riemsdijk Frank Dignum
John-Jules Ch. Meyer

Institute of Information and Computing Sciences
Utrecht University, The Netherlands
{ mehdi,birna,dignum,jj}@cs.uu.nl

Abstract

This paper presents the specification of a programming language for cognitive agents. This programming language is an extension of 3APL (An Abstract Agent Programming Language) and allows the programmer to implement agents' mental attitudes like beliefs, goals, plans, and actions, and agents' reasoning rules by means of which agents can modify their mental attitudes. The formal syntax and semantics of this language is presented.

1 Introduction

In research on agents, besides architectures, the areas of agent theories and agent programming languages are distinguished. Theories concern descriptions of (the behavior of) agents. Agents are often described using logic [9]. Concepts that are commonly incorporated in such logics are for instance knowledge, beliefs, desires, intentions, commitments, goals and plans.

It has been argued in the literature that it can be useful to analyze and specify a system in terms of these concepts [2, 7]. If the system would however then be implemented using an arbitrary programming language, it will be difficult to verify whether it satisfies its specification: if we cannot identify what for instance the beliefs, desires and intentions of the system are, it will be hard to check the system against its specification expressed in these terms. This is referred to by Wooldridge as the problem of ungrounded semantics for agent specification languages [11]. It will moreover be more difficult to go from specification to implementation if there is no clear correspondence between the concepts used for specification and those used for implementation.

To support the practical development of intelligent agents, several programming languages have thus been introduced that incorporate some of the concepts from agent logics. First there is a family of languages that use actions as their starting point to define commitments (Agent-0, [8]), intentions (AgentSpeak(L), [6]) and goals (3APL, [3]). In the language GOAL ([4]) on the other hand, the concept of declarative goals is central. In [10], the language Dribble was proposed which constitutes a synthesis between the declarative and the action centred approaches. Dribble is however a propositional language without variables, which

severely limits its programming power. In this paper, we propose an extension of the language 3APL, inspired by Dribble, with declarative goals *and* first order features. Furthermore, whereas in Dribble one can use goals for plan selection only, in this extension of 3APL we add rules for reasoning with goals. We will refer to the extension of 3APL presented in this paper, simply with the same name 3APL.

In the next section we introduce the syntax of the extended version of 3APL and indicate some of the important (new) features. In section 3 we describe the operational semantics of 3APL using state transitions. We give some conclusions and areas for further research in section 4.

2 Syntax

2.1 Beliefs and goals

The *beliefs* of a 3APL agent describe the situation the agent is in. The belief base can contain information the agent believes about the world and it can contain information that is internal to the agent. The *goals* of the agent on the other hand, denote the situation the agent wants to realize. The beliefs and goals are represented by a standard first-order language with variables (VAR), functions, predicates and the usual connectives. Terms (T_L) are built from variables and functions. Formulae are built from predicates, terms, negation of atomic formula, and conjunction of formulae. We assume that all variables are universally quantified with maximum scope. We will refer to this language as the *domain language (L)* in this paper, with typical formula ϕ.

A 3APL agent has a belief base and a goal base, both being sets of formulas from this domain language. In the rules which will be defined in the sequel, one needs to be able to refer to sentences from the belief base or goal base. Therefore, we define the following belief language and goal language on top of the domain language.

Definition 1 *(beliefs and goals) Let L be the domain language. Then, the belief language L_B and the goal language L_G are defined as follows:*
- if $\phi \in L$, then $\mathbf{B}\phi \in L_B$ and if $\beta, \beta' \in L_B$, then $\beta \wedge \beta' \in L_B$
- if $\phi \in L$, then $\mathbf{G}\phi \in L_G$ and if $\kappa, \kappa' \in L_G$, then $\kappa \wedge \kappa' \in L_G$

The belief and goal languages do not include negation. The main reason is to avoid a number of the problems occurring with substitutions and unifications concerning negation. They are very similar to those occurring in logic programs with negation. Although some (partial) solutions thus could be used from that area they would complicate the current paper unnecessary. We therefore left negation as a separate issue to be dealt with in a subsequent paper.

2.2 Plans

In order to reach its goals, a 3APL agent adopts *plans*. A plan is a sequence built from basic elements. The basic elements can be *basic actions*, *tests* on the belief base or *abstract plans* (sometimes called achievement goals [3]). As in the

languages GOAL and 3APL, basic actions specify the capabilities with which an agent should achieve a certain state of affairs. The effect of execution of a basic action is a change in the belief base of the agent. An abstract plan cannot be executed directly in the sense that it updates the belief base of an agent. Abstract plans serve as an abstraction mechanism like procedure calls in imperative programming. If a plan consists of an abstract plan, the abstract plan could be replaced by basic actions through reasoning rules. Basic actions and abstract plans consist of a name and a number of parameters that are terms from T_L.

Definition 2 *(plans) Assume a set of basic actions ACT and a set of abstract plans AP. Let $\beta \in L_B$. The plan language L_P consists of the following elements: the empty plan $E \in L_P$, $ACT \subseteq L_P$, $\beta? \in L_P$, $AP \subseteq L_P$ and if $\pi_1, \pi_2 \in L_P$, then $\pi_1; \pi_2$, `if` β `then` π_1 `else` π_2 `fi` and `while` β `do` π_1`od` $\in L_P$. We identify $E; \pi$ with $\pi; E$ and π.*

2.3 Reasoning Rules

We propose various rules to reason with goals, plans, and their interactions. These rules are conditionalized by beliefs.

Definition 3 *(rules) Let $\beta \in L_B$, $\kappa, \kappa_h, \kappa_b \in L_G$, and $\pi, \pi_h, \pi_b \in L_P$. We define sets of reasoning rules for goals GR, plans PR, and their interactions IR as follows:*
- $\kappa_h \leftarrow \beta \mid \kappa_b \in GR$,
- $\kappa \leftarrow \beta \mid \pi \in IR$,
- $\pi_h \leftarrow \beta \mid \pi_b \in PR$.

The *goal rules* are used to revise, generate or drop goals. For example, the goal rule $\mathbf{G}(on(x,y)) \leftarrow \mathbf{B}(tooHeavy(x) \wedge notHeavy(z)) \mid \mathbf{G}(on(z,y))$ can be used to revise one of an agent's goals: it informally means that if the agent desires to have block x on block y, but it believes that x is too heavy while z is not heavy, then it should revise its goal and aim to have block z on block y.

The *interaction rules* are used to generate plans to achieve goals. They are similar to the goal rules of Dribble. For example, the interaction rule $\mathbf{G}(on(x,z)) \leftarrow \mathbf{B}(on(x,y)) \mid move(x,y,z)$ states that if the agent desires to have block x on block z, but it believes that x is on block y, then it plans to move x from y and put it on z. The belief condition thus indicates when the plan could be selected to achieve the specified goal. Interaction rules can also be used to model reactive behavior in the sense that goals and actions are generated on the basis of agent's beliefs (about the environment) rather than on basis of existing goals. Reactive behavior can be modelled by interaction rules of the form $\top \leftarrow \beta \mid \pi$.

Finally, the *plan rules*, which are similar to the practical reasoning rules of 3APL, are used to revise and drop plans. For example, the plan rule $move(x,y,z) \leftarrow \mathbf{B}(\neg clear(x)) \mid on(u,x)?; move(u,x,Fl); move(x,y,z)$ informally means that if the agent plans to move block x from block y onto block z (i.e. $move(x,y,z)$), but it cannot move x because (it believes that) there is a block on x (i.e. $\mathbf{B}(\neg clear(x))$), then the agent should revise its plan by finding out which block (u) is on x (i.e.

$on(u, x)$?), moving u onto the floor (i.e. $move(u, x, Fl)$), and finally moving x from y onto z (i.e. $move(x, y, z)$). Plan rules are not used to generate plans. This is because we assume that a plan is only generated to achieve a certain goal. The generation of plans is therefore restricted to the interaction rules.

2.4 A 3APL configuration and agent

Above, the beliefs, goals and plans of a 3APL agent were defined. These are the elements that change during the execution of the agent. Together with a fourth *substitution* component, they constitute a 3APL configuration. The substitution is used to store values or bindings associated with first order variables. In the sequel, we will use $Free(e)$ to indicate the set of free variables (not bound by a quantifier) that occur in the syntactic element e and $dom(\theta)$ to denote the set of variables that form the domain of the substitution θ.

Definition 4 *(configuration) A configuration of a 3APL agent is a tuple $\langle \sigma, \gamma, \Pi, \theta \rangle$, where $\sigma \subseteq L$ is the belief base of the agent, $\gamma \subseteq L$ is the goal base of the agent, $\Pi \subseteq L_P \times L_G$ is the plan base of the agent[1], and θ represents the substitution that binds domain variables to domain terms, i.e. $\theta \subseteq \{ [x_i/t_i] \mid x_i \in VAR\ ,\ t_i \in T_L, x_i \notin Free(t_j), \forall i \neq j\ x_i \neq x_j \}$. Moreover, the belief and goal bases are assumed to be grounded, i.e. they consist of formulae in which no free variables occur. Finally, the belief base and the goal base of agents are such that for any goal $\phi \in \gamma$ the following holds: $\sigma \not\models \phi$, $\phi \not\models \bot$ and $\sigma \not\models \bot$.*

In this definition, we have defined Π as consisting of plan-goal formula pairs. The goal for which a plan is selected is recorded with the plan, because this for instance provides the possibility to drop a plan of which the goal is reached. Furthermore, goals may be revised or dropped and one might want to remove a plan associated with a goal which has been dropped, from the plan base.

Definition 5 *(agent) A 3APL agent is a tuple $\langle \sigma_0, \gamma_0, GR, IR, PR \rangle$ where $\langle \sigma_0, \gamma_0, \emptyset, \emptyset \rangle$ is the initial configuration, GR is a set of goal rules, IR is a set of interaction rules and PR is a set of plan rules.*

3 Semantics

We define an operational semantics for 3APL in terms of a transition system ([5]). A transition system is a set of derivation rules for deriving transitions. A transition is a transformation of one configuration into another and it corresponds to a single computation step. In order to define the semantics of the various rules, we first need to define the semantics of the belief and goal formulae.

Definition 6 *(semantics of belief and goal formulae) Let $\langle \sigma, \gamma, \Pi, \theta \rangle$ be an agent configuration, ϕ, $\phi' \in L$ and φ, $\varphi' \in L_B \cup L_G$.*

[1] Note that the (initial) goal that should be achieved by a plan is associated to that plan.

$$\begin{array}{lcl}\langle\sigma,\gamma,\Pi,\theta\rangle \models \mathbf{B}\phi & \Leftrightarrow & \sigma \models \phi \\ \langle\sigma,\gamma,\Pi,\theta\rangle \models \mathbf{G}\phi & \Leftrightarrow & \exists\phi' \in \gamma : \phi' \models \phi \textit{ and } \sigma \not\models \phi \\ \langle\sigma,\gamma,\Pi,\theta\rangle \models \varphi \wedge \varphi' & \Leftrightarrow & \langle\sigma,\gamma,\Pi,\theta\rangle \models \varphi \textit{ and } \langle\sigma,\gamma,\Pi,\theta\rangle \models \varphi'\end{array}$$

As explained above, the semantics of $\mathbf{G}\phi$ is defined in terms of separate goals, as opposed to defining it in terms of the entire goal base. The idea is, that all logical consequences of a particular goal are also goals, but only if they are not believed ([4]).

In the following, a set of derivation rules is proposed that specifies the semantics of various ingredients of 3APL. These rules specify the semantics of a 3APL agent with a set of goal rules GR, a set of plan rules PR and a set of interaction rules IR. We assume the usual notions of ground substitutions and application of a substitution as also defined in [3].

The first derivation rule specifies the execution of the plan base of a 3APL agent. The plan base of the agent is a set of plan-goal pairs. This set can be executed by executing the plan of one of the constituent plans-goal pairs.

Definition 7 *(plan base execution) Let* $\Pi = \{(\pi_1,\kappa_1),\ldots,(\pi_i,\kappa_i),\ldots,(\pi_n,\kappa_n)\} \subseteq L_P \times L_G$ *and* $\Pi' = \{(\pi_1,\kappa_1),\ldots,(\pi'_i,\kappa_i),\ldots,(\pi_n,\kappa_n)\} \subseteq L_P \times L_G$ *be plan bases,* θ,θ' *be ground substitutions, and* $V = Free(\Pi)$*. Then, the derivation for the execution of a set of plans is specified in terms of the execution of individual plans as follows:*

$$\frac{\langle\sigma,\gamma,(\pi_i,\kappa_i),\theta\rangle_V \rightarrow \langle\sigma',\gamma',(\pi'_i,\kappa_i),\theta'\rangle}{\langle\sigma,\gamma,\Pi,\theta\rangle \rightarrow \langle\sigma',\gamma',\Pi',\theta'\rangle}$$

Transitions for individual plans are parameterized by the set of free variables V of the entire plan base Π. This is necessary because in the transition rules for individual plans, sometimes reference needs to be made to this set. In the following, we use *Goal* as a function of type $L_G \rightarrow \wp(L)$ that removes the $\mathbf{G}$ modalities from a goal formula returning its goals from L. For example, $Goal(\mathbf{G}(p(a)) \wedge \mathbf{G}(q(b))) = \{p(a), q(b)\}$.

Now we will introduce the derivation rules for the execution of basic plan elements: basic actions and tests. We do not introduce derivation rules for abstract plans, because abstract plans cannot be executed. They can only be transformed using plan rules.

Definition 8 *(basic action execution) Let* $\alpha \in ACT$ *and let* $\mathcal{T}$ *be a function that specifies the belief update resulting from the execution of basic actions, then the execution of a single action is specified as follows:*

$$\frac{\mathcal{T}(\alpha\theta,\sigma) = \sigma' \;\&\; \langle\sigma,\gamma,\{(\alpha,\kappa)\},\theta\rangle \models \kappa}{\langle\sigma,\gamma,(\alpha,\kappa),\theta\rangle_V \rightarrow \langle\sigma',\gamma',(E,\kappa),\theta\rangle}$$

where $\gamma' = \gamma \backslash \{\phi \in \gamma \mid \; \phi \in Goal(\kappa) \;\&\; \sigma' \models \phi\}$.

The condition $\langle\sigma,\gamma,\{(\alpha,\kappa)\},\theta\rangle \models \kappa$ guarantees that the action can only be executed if the goal for which α was selected, is still entailed by the current configuration. This condition could be removed, leaving the decision of whether to enforce

it, up to a meta-program called the deliberation cycle ([1]). The substitution θ is used to instantiate free variables in the basic action α.

The derivation rule for the execution of the test can bind the free variables that occur in the test formula for which no bindings have been computed yet.

Definition 9 *(test execution) Let $\beta \in L_B$ and let τ be a ground substitution such that $dom(\tau) = Free(\beta\theta)$, then*

$$\frac{\langle \sigma, \gamma, \Pi, \theta \rangle \models \beta\theta\tau}{\langle \sigma, \gamma, (\beta?, \kappa), \theta \rangle_V \rightarrow \langle \sigma, \gamma, (E, \kappa), \theta\tau \rangle}$$

The derivation rule for the execution of sequential composition is defined in the standard way below. For reasons of space, we leave out the rule for the if-then-else and while construct.

Definition 10 *(execution of sequential composition)*

$$\frac{\langle \sigma, \gamma, (\pi_1, \kappa), \theta \rangle_V \rightarrow \langle \sigma', \gamma', (\pi_1', \kappa), \theta' \rangle}{\langle \sigma, \gamma, (\pi_1; \pi_2, \kappa), \theta \rangle_V \rightarrow \langle \sigma', \gamma', (\pi_1'; \pi_2, \kappa), \theta' \rangle}$$

The goal associated with some plan is passed on unchanged through the transitions modifying this plan.

We will now define the transition rules for the reasoning rules. For this, we will need the notion of a variant. A syntactic element e is a variant of another element e' in case e can be obtained from e' by renaming of variables. We will use variants of rules to avoid unwanted bindings between variables in those rules and variables in the plan base (V) or in $dom(\theta)$.

A goal rule $\kappa_h \leftarrow \beta \mid \kappa_b$ is applicable if its head is derivable from the agent's goal base and its condition is derivable from the agent's belief base. The application of the goal rule only affects the goal base of the agent.

Definition 11 *(goal rule application) Let $\kappa_h, \kappa_b \in L_G$, $\beta \in L_B$, η, τ be ground substitutions such that $dom(\eta) = Free(\kappa_h)$ and $dom(\tau) = Free(\beta\eta)$. Let the rule $\kappa_h \leftarrow \beta \mid \kappa_b$ be a variant of a goal rule $\in GR$ such that no free variables in the rule occur in $dom(\theta)$ and $Free(\kappa_b) \subseteq Free(\kappa_h) \cup Free(\beta)$. Then the transition rule for the goal rule $\kappa_h \leftarrow \beta \mid \kappa_b$ is defined as follows:*

$$\frac{\langle \sigma, \gamma, \Pi, \theta \rangle \models \kappa_h\eta \;\&\; \langle \sigma, \gamma, \Pi, \theta \rangle \models \beta\eta\tau \;\&\forall \phi'' \in Goal(\kappa_b) : \sigma \not\models \phi''\eta\tau}{\langle \sigma, \gamma, \Pi, \theta \rangle_V \rightarrow \langle \sigma, \gamma', \Pi, \theta \rangle}$$

where $\gamma' = (\gamma \backslash \{\phi \in \gamma \mid \phi' \in Goal(\kappa_h)$ and $\phi \models \phi'\eta\}) \cup \{\phi''\eta\tau \mid \phi'' \in Goal(\kappa_b)\}$.

The effect of the application of the goal rule on the goal base is that it removes goals of which a goal in the head of the rule is a logical consequence (for all substitutions τ) from the goal base, and adds the goals in the body of the goal rule to the goal base (for any substitutions τ and η).

A plan rule $\pi_h \leftarrow \beta \mid \pi_b$ is applicable if its head π_h unifies with a plan of the agent and its condition β is derivable from agent's beliefs. We assume that the revised plan π_b is designed to achieve the same goal. Therefore, the goal associated with plan π_h in the plan base will be associated with the revised plan π_b as well. The application of a plan rule only affects the plan base of the agent.

Definition 12 *(plan rule application) Let* $\pi_h, \pi_b \in L_P, \kappa \in L_G, \beta \in L_B$, $\pi_h \leftarrow \beta \mid \pi_b$ *be a variant of a plan rule* $\in PR$ *such that no free variables in the rule occur in* V *or* $dom(\theta)$, η *be a most general unifier for* π *and* π_h, *and* θ, τ *be ground substitution such that* $dom(\tau) = Free(\beta\eta)$. *Then,*

$$\frac{\langle \sigma, \gamma, \Pi, \theta \rangle \models \beta\eta\tau \ \&\ \langle \sigma, \gamma, \{(\alpha, \kappa)\}, \theta \rangle \models \kappa}{\langle \sigma, \gamma, (\pi, \kappa), \theta \rangle_V \rightarrow \langle \sigma, \gamma, (\pi_b\eta\tau, \kappa), \theta \rangle}$$

The effect of the application of the plan rule on the plan base is that the plan π is replaced by the body π_b of the plan rule instantiated with the substitution η, that resulted from matching the head of the rule with the revised plan, and with the substitution τ that resulted from matching the condition of the rule with the belief base.

An interaction rule $\kappa \leftarrow \beta \mid \pi$ specifies that the goal κ can be achieved by plan π if β is derivable from the agent's beliefs. An interaction rule only affects the plan base of the agent.

Definition 13 *(interaction rule application) Let* $\kappa \in L_G$, $\beta \in L_B$, $\pi \in L_P$, $\kappa \leftarrow \beta \mid \pi$ *be a variant of an interaction rule* $\in IR$ *such that no free variables in the rule occur in* V *or* $dom(\theta)$, *and* η, τ *be ground substitutions such that* $dom(\eta) = Free(\kappa)$ *and* $dom(\tau) = Free(\beta\eta)$. *Then,*

$$\frac{\langle \sigma, \gamma, \Pi, \theta \rangle \models \kappa\eta \ \&\ \langle \sigma, \gamma, \Pi, \theta \rangle \models \beta\eta\tau}{\langle \sigma, \gamma, \Pi, \theta \rangle_V \rightarrow \langle \sigma, \gamma, \Pi \cup \{(\pi\eta\tau, \kappa\eta)\}, \theta \rangle}$$

Note that the goal $\kappa\eta$ that should be achieved by the plan $\pi\eta\tau$ is associated with it. It is only this rule that associates goals with plans. The goal base of the agent does not change because the plan $\pi\eta\tau$ is not executed yet; the goals of agents change only after execution of plans.

The semantics of a 3APL agent is derived directly from the transition relation $\rightarrow$. The meaning of a 3APL agent consists of a set of so called computation runs. A computation run $\texttt{CR}(s_0)$ for a 3APL agent is a finite or infinite sequence $s_0, \ldots, s_n$ or $s_0, \ldots$ where s_i are configurations, and $\forall_{i>0}: s_{i-1} \rightarrow s_i$ is a transition in the transition system for the 3APL agent. The meaning of a 3APL agent $\langle \sigma_0, \gamma_0, GR, PR, IR \rangle$ is the set of computation runs $\texttt{CR}(\langle \sigma_0, \gamma_0, \emptyset, \emptyset \rangle)$. Note that the first state of the computation runs is the initial mental state of the 3APL agent.

4 Conclusion and Future Research

In this paper we have described the syntax and semantics of an agent programming language that includes some of the classical elements of the theory of agents, i.e. beliefs, goals and plans. We thus conjecture that it should be possible to verify whether a 3APL program satisfies a given specification in terms of beliefs, goals and plans. An interpreter for the basic form of 3APL is already implemented and extensions are currently being programmed. The interpreter will enable us to evaluate the effectiveness of the language for problems of realistic complexity.

Other classical elements of agent theories such as knowledge, desires, intentions, and commitments are not considered in this work and left for future research. We believe that an implementation language that provides programming constructs for a full range of elements involved in agent theories facilitates direct and easy implementation of agent specifications.

References

[1] M. Dastani, F. de Boer, F. Dignum, and J.-J. Meyer. Programming agent deliberation: An approach illustrated using the 3apl language. In *Proceedings of The Second Conference on Autonomous Agents and Multi-agent Systems (AAMAS'03)*, Melbourne, Australia, 2003.

[2] D. Dennet. *The intentional stance*. The MIT Press, Cambridge, 1987.

[3] K. Hindriks, F. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3APL. *Int. J. of Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[4] K. Hindriks, F. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming with declarative goals. In N. Jennings and Y. Lesperance, editors, *Intelligent Agents VI - Proceedings of ATAL'2000*, LNAI-1757. Springer, Berlin, 2001.

[5] G. Plotkin. A structural approach to operational semantics. Technical report, Aarhus University, Computer Science Department, 1981.

[6] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. van der Velde and J. Perram, editors, *Agents Breaking Away (LNAI 1038)*, pages 42–55. Springer-Verlag, 1996.

[7] A. S. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319, San Francisco, CA, June 1995.

[8] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[9] W. van der Hoek, B. van Linder, and J.-J. Ch. Meyer. An integrated modal approach to rational agents. In M. Wooldridge and A. Rao, editors, *Foundations of Rational Agency*, Applied Logic Series 14, pages 133–168. Kluwer, Dordrecht, 1998.

[10] M. B. van Riemsdijk, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in Dribble: from beliefs to goals with plans. In *Proceedings of AAMAS*, Melbourne, Australia, July 2003.

[11] M. Wooldridge. *An introduction to multiagent systems*. John Wiley and Sons, LTD, West Sussex, 2002.

Decisions, Deliberation, and Agent Types

Mehdi Dastani [a] Leendert van der Torre [b]

[a]IICS, Utrecht University, mehdi@cs.uu.nl
[b]CWI Amsterdam, torre@cwi.nl

Abstract

In this paper we investigate the relation between decisions, deliberation and agent types. In particular, we are interested how deliberation leads to decisions, and how agent types classify patterns of deliberation. We therefore consider Classical and Qualitative Decision Theories (CDT and QDT), the Beliefs-Desire-Intention (BDI) model, 3APL systems, and Belief-Obligation-Intention-Desire (BOID) systems. The first two are based on a decision rule which expresses a notion of rationality, whereas the latter three are based on deliberation processes and agent types.

1 Introduction

In recent years the interest in models of decision making for autonomous agents has increased. Classical decision theory [13] explains the decision making behavior in terms of a probability distribution and a utility function together with a decision rule, but other approaches criticize the representation of classical decision theory as being non-practical and unrealistic. According to them it is hard to translate all factors that influence the decision making behavior of an agent in terms of two functions that assign numbers to actions and states. In contrast, they aim at explaining the decision making behavior of agents in terms of qualitative concepts such as preference and likelihood ordering, or cognitive concepts such as beliefs, desires, intentions, and obligations.

There are many conceptualizations and formalizations of decision making. In [6] we compare classical decision theory with qualitative decision theory, knowledge-based systems and belief-desire-intention models developed in artificial intelligence and agent theory. They all contain representations of information and motivation. Examples of informational attitudes are probability distributions, qualitative abstractions of probabilities, knowledge, and beliefs. Examples of motivational attitudes are utility functions, qualitative abstractions of utilities, goals, and desires. Each of them encodes a set of alternatives to be chosen from. This ranges from a small predetermined set, a set of decision variables, through logical formulas, to branches of a tree representing events through time. Moreover, they have a way of formulating how a decision is made. Classical and qualitative decision theory focus on the optimal decisions represented by a decision rule. Knowledge-based systems and belief-desire-intention models focus on a model of the representations used in decision making, inspired by cognitive notions like belief, desire, goal and intention. Relations among these concepts express an agent

type, which constrains the deliberation process. In this paper we are interested in the relation between decisions, deliberation and agent types. In particular, we are interested in the question:

How does deliberation based on agent types lead to decisions?

This question breaks down in three sub-questions:

1. What is an agent decision? To answer this question we discuss different approaches and concepts used to explain the decision making behavior of agents. The first is classical decision theory [8, 13] and the second is qualitative extension (QDT) [1, 9].

2. What is agent deliberation and how does it lead to a decision? The third approach we discuss is based on an abstract model of the mental attitudes of an agent: beliefs, desires and intentions (BDI) [2, 4, 10]. The fourth (3APL) introduces the deliberation process that determines which actions should be performed for a given set of underlying cognitive concepts.

3. How do agent types classify patterns of deliberation? The fifth approach we discuss to answer this question is also based on mental attitudes extended with obligations (BOID) [3].

The layout of this paper is as follows. We first discuss classical and qualitative decision theory, thereafter agent types in the BDI approach and the BOID approach, and finally we focus on deliberation in 3APL systems.

2 Decisions

In classical decision theory, a decision is a choice made by a decision maker of an action from a set of alternatives. A decision is good if it is an alternative that the decision maker believes will prove at least as good as other alternative actions. Good decisions are formally characterized as actions that maximize expected utility, a notion involving both belief and goodness [7].

Definition 1 *Let A stand for the set of actions or alternatives. With each action, a set of outcomes is associated. Let W stand for the set of all possible worlds or outcomes. Let U be a measure of outcome value that assigns a utility $U(w)$ to each outcome $w \in W$, and let P be a measure of the probability of outcomes conditional on actions, with $P(w|a)$ denoting the probability that outcome w comes about after taking action $a \in A$ in the situation under consideration. The expected utility $EU(a)$ of an action a is the average utility of the outcomes associated with the alternative, weighing the utility of each outcome by the probability that the outcome results from the alternative, that is, $EU(a) = \sum_{w \in W} U(w)P(w|a)$. A rational decision maker always maximizes expected utility.*

Qualitative decision theory relaxes the assumption of classical decision theory that the decision making agent is able to weigh all possible alternative courses

of action before choosing one of them. In realistic situations decision making agents are resource bounded in the sense that they have partial information and do not have resources to compare the utility of possible states. The research on qualitative decision theory aims therefore to develop representation and reasoning schemes for partial information and generic preferences to represent probabilities of states and generic preferences over those states [7]. Typically qualitative orderings are introduced that represent the likelihood (probability) and desirability (utility) of states. In contrast to the classical decision theory where a decision rule such as maximum expected utility determines the course of actions, in qualitative decision theory, the course of actions are decided based on various strategies such as maximize potential gain or minimize potential loss [1].

Boutilier [1] proposes a logic and possible worlds semantics for representing and reasoning with qualitative probabilities and utilities, and suggests several strategies for qualitative decision making based on this logic. His semantics is not quantitative (like CDT), but purely qualitative. Consequently, the maximum expected utility (MEU) decision rule is replaced by qualitative rules like Wald's criterion. The conditional preference is captured by a preference ordering (an ordinal value function) that is defined on possible worlds. The preference ordering represents the desirability of worlds. Similarly, probabilities are captured by an ordering, called normality ordering, on possible worlds representing their likelihood.

Definition 2 *The possible worlds semantics for this logic is based on models of the form* $M = \langle W, \leq_P, \leq_N, V \rangle$ *where* W *is a set of worlds (outcomes),* $\leq_P$ *is a transitive and connected preference ordering relation on* W*,* $\leq_N$ *is a transitive and connected normality ordering relation on* W*, and* V *is a valuation function.*

In Boutilier's logic, conditional preferences can be represented by means of ideal operator I. We have that a model M satisfies the formula $I(\varphi|\psi)$ if the preferred or best or minimal ψ worlds are φ worlds. For example, let u be the proposition 'agent has umbrella' and r be the proposition 'it rains', then $I(u|r)$ expresses that in the most preferred rain-worlds the agent has an umbrella. Similarly, probabilities are represented in this logic by means of a normative conditional connective $\Rightarrow$. For example, let w be the proposition 'the agent is wet' and r be the proposition 'it rains', then $r \Rightarrow w$ expresses that the agent is wet at the most normal rain-worlds. A rational agent is then assumed to attempt to achieve the best possible world consistent with its (default) knowledge. Based on this idea, the notion of goals is introduced as being some proposition that the agent desire to make true. Boutilier assumes (for simplicity of presentation) that Cl(KB) is finitely specifiable and takes it to be a single propositional sentence.

Definition 3 *Given a set of facts* KB*, a goal is thought to be any proposition* α *such that* $M \models I(\alpha \mid Cl(KB))$ *where* $Cl(KB)$ *is the default closure of the facts* KB *defined as follows:* $Cl(KB) = \{\alpha \mid KB \Rightarrow \alpha\}$.

The expressions of this logic represent qualitative constraints such that the proposed logic enables the agents to reason directly with such constraints.

3 Agent types

In BDI systems, the (partial) information on states is reduced to dichotomous values (0-1); the propositions are believed or not. This abstraction is called the beliefs of the decision making agent. Similarly, the (partial) information about the objectives of the decision making agent is reduced to dichotomous values (0-1); the propositions are desired or not. This abstraction is called the desires of the decision making agent. Finally, the (partial) information about the previous decisions, to which the agent is still committed to, is represented by dichotomous values (0-1); the proposition are intended or not.

The consequence of the fact that we no longer have pre-orders, like in Boutilier's logic, but only an unstructured set, is that we can now only represent monadic expressions like $B(\varphi)$ and $D(\varphi)$, no longer dyadic expressions like $I(\varphi|\psi)$. We have that a world (at a certain time point) of the model satisfies $B(\varphi)$ if φ is true in all accessible worlds (at the same time point). Since there is no such ordering on the possible worlds in BDI, each desire world can in principle be chosen as the goal world. BDI proposes the notion of agent types which can be considered as abstract decision patterns. The most famous ones are the realism and the commitment strategies defined in BDI systems [11].

The realism constraint states that agent's desire should be consistent with its beliefs. Formally, the realism axiom states that the set of desire accessible worlds should be a subset of the set of belief accessible worlds, $B(\varphi) \rightarrow D(\varphi)$. Moreover, the belief and desire worlds should have identical branching time structure (the same alternatives), $\forall w \forall t \forall w'$ if $w' \in \mathcal{D}_t^w$ then $w' \in \mathcal{B}_t^w$, where $\mathcal{D}_t^w$ ($\mathcal{B}_t^w$) is the set of desire (belief) accessible worlds from the world w at time t. The definition of realism includes an additional axiom to reduce the set of desire worlds and to guarantee that chosen desire world is consistent with the worlds that are already chosen as the goal worlds. Formally, this axiom states that the intention accessible worlds should be a subset of desire accessible worlds, i.e., $D(\varphi) \rightarrow I(\varphi)$. Moreover, the desire and intention worlds should have identical branching time structure (the same alternatives), $\forall w \forall t \forall w'$ if $w' \in \mathcal{I}_t^w$ then $w' \in \mathcal{D}_t^w$.

The commitment strategies are different types of constraints resulting in additional agent types. These axioms determine when intentions or previously decided goals should be reconsidered or dropped. In BDI, choosing to drop a goal is thus considered to be as important as choosing a new goal. These constraints, called commitment strategies, involve time and intentions and express the dynamics of decision making. The well-known commitment strategies are 'blindly committed decision making', 'single-minded committed decision making', and 'open-minded committed decision making'. For example, the single-minded commitment strategy states that an agent remains committed to its intentions until either it achieves its corresponding objective or does not believe that it can achieve it anymore.

In the BOID approach [3] goals are generated based on the derivation of so-called extensions in default logic [12]. Default logic extends the inference rule modus ponens with two new mechanisms. First, there is a consistency constraint on the inference process, such that rules are only applied if they do not lead to an inconsistency. Second, the application of defeasible rules may result in conflicting

outputs resulting alternative sets of logic formulas. In BOID, goal generation is based on prioritized default logic. The specification of goal generation process - our instantiation of a default theory - contains the specification of a set of facts, a set of belief rules, a set of obligation rules, a set of intention rules, a set of desire rules, and the specification of a priority function ρ on the rules.

In BOID agent types are defined in terms of goal generation process. These agent types are based on overriding, such that in realistic agents beliefs override other mental attitudes, and in social agents obligations override desires. In particular, agent types based on goal generation are conflict resolution methods. An agent has a conflict if the goal generation process derives multiple extensions. A conflict is resolved if the priority function is adapted such that no alternative extensions are generated. A mental attitude conflicts with another mental attitude if two rules from different attitudes are applicable, but applying both leads to an inconsistent set. A rule overrides another rule if it has a higher priority. Finally, agent types are formalized in Definition 4 as constraints on the set of available priority functions. When the goal generation process starts the selected priority function obeys the constraints corresponding to the agent type.

Definition 4 *An agent type based on goal generation is a consistent set of constraints on priority function ρ of the form $X \succ Y$ with $X, Y \in \{B, O, I, D\}$ defined as follows: $\forall rule\_x \in X \ \forall rule\_y \in Y \ \rho(rule\_x) > \rho(rule\_y)$. A primitive agent type contains a single constraint. A complete agent type is a maximal consistent set of constraints.*

There are twelve primitive agent types, which are listed below together with the corresponding constraint. They are ordered in six pairs, each agent type A together with its inverse $\{X \succ Y \mid Y \succ X \in A\}$.

Constraint	Agent type
$B \succ O$ $(O \succ B)$	Realistic with respect to obligations (dogmatic)
$B \succ I$ $(I \succ B)$	Realistic with respect to intentions (over-committed)
$B \succ D$ $(D \succ B)$	Realistic with respect to desires (wishful thinker)
$O \succ I$ $(I \succ O)$	(Un)Stable with respect to obligations
$O \succ D$ $(D \succ O)$	Social (selfish)
$I \succ D$ $(D \succ I)$	(Un)Stable with respect to desires

An agent type is a set of primitive agent types. For example, the realistic agent type is $\{B \succ O, B \succ I, B \succ D\}$, the stable agent type is $\{I \succ O, I \succ D\}$, the social stable agent type is $\{I \succ O, I \succ D, O \succ D\}$, etc. Moreover, agent types can be derived. For example, since orderings are transitive we can derive that an agent which is unstable with respect to obligations ($O \succ I$) and stable with respect to intentions ($I \succ D$) is social ($O \succ D$). There are twenty-four complete agent types, of which the six realistic ones are listed below.

Constraints	Agent type
$B \succ O \ O \succ I \ I \succ D$	Realistic, unstable-O, stable-D, social
$B \succ O \ O \succ D \ D \succ I$	Realistic, unstable-O, unstable-D, social
$B \succ I \ I \succ O \ O \succ D$	Realistic, stable-O, stable-D, social
$B \succ I \ I \succ D \ D \succ O$	Realistic, stable-O, stable-D, selfish
$B \succ D \ D \succ O \ O \succ I$	Realistic, unstable-O, unstable-D, selfish
$B \succ D \ D \succ I \ I \succ O$	Realistic, stable-O, unstable-D, selfish

4 Deliberation language

In order to specify deliberation processes, a deliberation language [5] is developed whose expressions specify various deliberation processes. The deliberation language is a many-sorted meta-language and its sorted terms refer to various types of expressions of an object-level language. The object-level language specifies cognitive agents in terms of their beliefs, goals, plans, reasoning rules, etc. The deliberation language is imperative and set-based. It is imperative because it is used to program the flow of control and it is set-based because it is used to select goals and rules from the set of object-level goals and rules. The expressions of the deliberation language specify the selection and application mechanisms and the order in which decisions should be taken. The selection mechanism involves decisions such as which goals or plans should be selected and executed, which rules should be selected and applied, or which mental bases should be updated by a formula. The order of decisions determine whether goals should be updated before getting executed or should the plans be executed before selecting a new goal.

In particular, the deliberation includes sorted terms that refer to object-level entities such as goals, actions, plans, and rules, predicates that express relations between terms, and statements that selects goals, plans, reasoning rules, and generate plans or examine the consequences of the plans. In addition, the deliberation includes imperative language constructs to compose meta-level statements and specify how to process the object-level entities. For now it is important to illustrate only the statements of the deliberation language through which various types of deliberation activities can be implemented. The operational semantics of these statements are presented in [5].

Definition 5 *Let s be any sort, t_s be a term of sort s, and V_s be a variable of sort s. The set of meta-statements MS is defined as follows:*
- $V_s := t_s \in MS$
- $executegoal(t_{goal}, V_{goal})$,
  $selectrule(t_{goals}, t_{rules}, V_{goal}, V_{rule})$,
  $applyrule(t_{rule}, t_{goal}, V_{goal}) \in MS$
- $plan(t_{goal}, V_{plan})$,
  $replan(t_{goal}, t_{plans}, V_{plans})$,
  $executeplan(t_{plan})$,
  $backtrackplan(t_{goal}, t_{plan}, V_{plan}) \in MS$
- *if φ is a meta-formula, $\alpha, \beta \in MS$, then*
  $\alpha ; \beta$,
  `IF` φ `THEN` α `ELSE` β,
  `WHILE` φ `DO` $\alpha \in MS$

The statements of the deliberation language can be divided into four classes: the assignment statement, the goal statements, the plan statements, and the composition statements. The assignment statement $V_s := t_s$ specifies the assignment of a sorted term t_s to a variable V_s of the same sort. The composition statements specify the sequential order of statements, conditional and iteration statements.

The statement $excutegoal(t_{goal}, V_{goal})$ specifies the execution of an individual

goal denoted by the term t_{goal}. As a goal may not be fully executable, some part of it may remain not executed. This remaining part is itself a goal which is assigned to the variable V_{goal}. The statement $selectrule(t_{goals}, t_{rules}, V_{goal}, V_{rule})$ specifies the selection of a rule and a goal from the set of rules denoted by the terms t_{rules} and the set of goals denoted by the term t_{goals}, respectively. The selected rule should be applicable to the selected goal. The selected rule and goal are assigned to variables V_{rule} and V_{goal}, respectively. The statement $applyrule(t_{rule}, t_{goal}, V_{goal})$ specifies the application of a rule denoted by the term t_{rule} to a goal denoted by the term t_{goal}. The resulting revised goal is assigned to variable V_{goal}.

The statement $plan(t_{goal}, V_{plan})$ specifies the generation of a plan V_{plan} to achieve goal t_{goal}. The statement $replan(t_{goal}, t_{plans}, V_{plan})$ specifies the generation of a new plan to replace plans t_{plans} that were generated to achieve goal t_{goal}. The new plan is assigned to variable V_{plan}. The statement $executeplan(t_{plan})$ specifies the execution of the plan denoted by the term t_{plan}. Finally, the last statement $backtrackplan(t_{goal}, t_{plan}, V_{plan})$ specifies the generation of a new plan by backtracking through the space of possible plan. Note that *replan* statement does not take the order of the plans in the plan space into consideration. The existing plan denoted by t_{plan}, which was generated to achieve the goal denoted by t_{goal}, is replaced by a new plan which is assigned to variable V_{plan}.

Various types of deliberation processes can be specified as expressions of this deliberation language. Because of the lack of space we refer to [5] for examples of such expressions.

5 Summary

We considered a range of systems, which are concerned with decisions, deliberation or agent types. We asked ourselves the following question:

> How does deliberation based on agent types lead to decisions?

This question breaks down in three sub-questions. First, what is an agent decision? To answer this question we discussed theories which define what a decision is, so-called decision theories. The first is classical decision theory [8, 13] and the second it qualitative extension (QDT) [1, 9]. In all these theories, a decision is the best option among a set of alternatives. We also illustrated that classical as well as recently proposed qualitative decision theories do not explain how decisions can be found, but in Boutilier's decision theory the concept of goal plays a central role. This is also a crucial concept in the three theories of deliberation studied in this paper.

Second, what is agent deliberation and how does it lead to a decision? To answer this question we discussed two other approaches. The first approach we discussed is based on an abstract model of the mental attitudes of an agent: beliefs, desires and intentions (BDI) [2, 4, 10]. The second (3APL) is similar to BDI except that the decision rules is replaced by a process called the deliberation process. It is this process that determines which actions should be performed for a given set of underlying cognitive concepts. The discussion on BDI systems and 3APL

illustrates how deliberation finds decisions. The deliberation process is based on cognitive concepts like beliefs, desires, goals, intentions and plans. The system generates goals, and thereafter finds plans to achieve these goals.

Third, how do agent types classify patterns of deliberation? The fifth approach we discussed to answer this question is also based on mental attitudes extended with obligations (BOID) [3]. The discussion on BDI-CTL and BOID illustrates how agent types classify deliberation. The deliberation process in BDI and BOID has been characterized in terms of agent types or deliberation patterns. In BDI-CTL the agent types represent how beliefs, goals and intentions are related, and when goals are maintained or dropped. In BOID the agent types represent how the agent resolves his conflicts. For example, a selfish agent lets its desires take precedence over his obligations. Using cognitive concepts and formulating the decision rule in terms of deliberation patterns makes the later two approaches cognitive theories of decision making.

Acknowledgments

Thanks to Jan Broersen, Zhisheng Huang and Joris Hulstijn for many discussions.

References

[1] C. Boutilier. Toward a logic for qualitative decision theory. In *Proceedings of KR'94*, pages 75–86. Morgan Kaufmann, 1994.

[2] M. Bratman. *Intention, plans, and practical reason*. Harvard University Press, Cambridge Mass, 1987.

[3] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.

[4] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42 (2-3):213–261, 1990.

[5] M. Dastani, F. de Boer, F. Dignum, and J.J. Meyer. Programming agent deliberation: An approach illustrated using the 3APL language. In *Proceedings of AAMAS'03*, 2003. ACM Press.

[6] M. Dastani, J. Hulstijn, and L. van der Torre. How to decide what to do? *European Journal of Operations Research*, to appear.

[7] J. Doyle and R. Thomason. Background to qualitative decision theory. *AI magazine*, 20:2:55–68, 1999.

[8] R. C. Jeffrey. *The Logic of Decision*. McGraw-Hill, New York, 1965.

[9] J. Pearl. From conditional ought to qualitative decision theory. In *Proceedings of UAI'93*, pages 12–20. John Wiley and Sons, 1993.

[10] A. Rao and M. Georgeff. Modeling rational agents within a bdi architecture. In *Proceedings of KR'91*, pages 473–484. Morgan Kaufmann, 1991.

[11] A. Rao and M. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8:293–342, 1998.

[12] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[13] L. Savage. *The foundations of statistics*. Wiley, New York, 1954.

Using Negative Emotions to Impair Game Play

Doug DeGroot and Joost Broekens

Leiden Institute of Advanced Computer Science

Leiden University, Leiden, The Netherlands

Abstract

Emotions make significant contributions to rational thought and behavior at multiple levels. Most research with computational models of emotion focuses on reactive behaviors caused by base or learned emotions; yet emotions can play a role in purposeful behaviors as well, leading to emotion-related actions rather than reactions. Higher-level cognitive processes can reflect on the emotional state, treating emotions as data rather than forces, and incorporate them into consciously selected behaviors, plans, and strategies. We have developed a computational framework for exploring cognitive decision processes that reason about emotional actions as an integral component of strategy and control over emotions. To validate this framework, we are implementing a prototype gaming system that exhibits conscious use of the emotional state and negative emotional behaviors during a game of chess in an attempt to impair its opponent's game play.

1. Introduction

There is increasing interest in the integration of models of computational emotions with models of cognition and behavior — especially within the arenas of cognitive neuroscience and intelligent virtual humans and agents — as it is becoming increasingly accepted that emotions comprise a significant component of rational thinking and human behavior. Most of the research with intelligent virtual humans, however, has focused on emotionally reactive, behavioral response mechanisms in an attempt to produce consistent, believable animations of virtual humans, including facial movements, gestures, voice, etc. In these models, the goal of computational emotions is to achieve increased "believability" of the agents, increased enjoyment from interacting with the agents, or to achieve a more meaningful, value-added communication channel between the human and the agent [1]. This is *communication-driven* emotion modeling [2]. Other research efforts have explored the use of emotional stimulus evaluation (appraisal) to maintain representations of an agent's emotional state. The state changes in relation to the agent's evaluation of external events compared to its internal desires and goals. The resulting emotional states are

used to influence the action decisions of the agent, to modify its beliefs, or to influence its reasoning and planning activities. This is *appraisal-driven* emotional modeling [2].

In essentially all these efforts, the main goal of incorporating emotions is achieving higher performance and/or higher satisfaction levels on the part of the human through increased believability and emotional intelligence of the agent. To make agents more predictable and understandable by the humans interacting with them, the focus is on the direct, positive correlation of emotions with behaviors, wherein positive (negative) emotions result in positive (negative) behaviors.

Things are a bit different in the computer gaming arena — but not much. Emotional modeling in games is also used primarily to make computer characters behave more "believably". Unlike intelligent virtual agent applications, though, computer games are often designed specifically to beat their human opponents. They attempt to reduce the human's performance level rather than increase it. However, even here, the focus has remained on positively correlated, believable behaviors.

2. A Shift to Negative Emotional Behaviors

We are exploring the utility of negatively correlated emotional behaviors on the part of a game-playing software robot in order to determine whether and how they might prove effective in emotionally impairing a human opponent's game play. We use computational emotions neither to improve the human's performance nor his satisfaction with the game; rather, our goal is achieving better game play on the part of the computer at the expense of the human.

Our research involves two fundamental shifts in focus from these other efforts. First, we integrate appraisal-driven with communication-driven emotion modeling. The effect is that the agent's own emotion-driven actions and behaviors loop back to influence its own desires, plans, and beliefs. Additionally, its conscious plans and desires can elicit actions on its part that are intended specifically to influence both its own emotional state and that of its human opponents. Our model thus embodies an "end-to-end" approach. This is similar to the approach of Franklin's Conscious-Mattie [6]. Second, unlike C-Mattie, we support negatively coupled emotions and behaviors, conscious, purposeful control of emotions, and dissimulation.

We have developed a general-purpose framework – the NTIM (for "intimidation") framework – that incorporates a computational model of emotions into a self-aware reasoning engine. The reasoning engine is capable of emotional-response control and purposeful selection of behavioral responses. We have implemented a prototype emotional, chess-playing agent based on this framework in order to explore the cognitive aspects of conscious control of

emotional behavior. We refer to the prototype as the NTIM game-playing robot below.

Rather than employing emotions in a direct, positive feedback loop to achieve behavioral homeostatis [3], our use of emotions is in expressing negative emotions and behaviors in an attempt to defeat the human more resoundingly. We strive to *disrupt* the emotional channel. This requires cognitive processing of emotions as data rather than behavioral forces. Specifically, our use of negative emotional behavior is intended to imbue within the human opponent a state of negative emotions that will impair the human's play, thereby reducing both his performance and his satisfaction levels. We believe this inverse use of computational emotions is novel.

3. Chess and Emotions

We selected Chess as the initial application within which to study our framework. There are several reasons for this choice. Foremost among these is that chess has stood as the quintessential game for showcasing artificial intelligence over the past 50 years. Numerous chess playing systems have been explored during this time, but all with a complete reliance on analytical capabilities for playing and winning the game. Emotions have not yet played a role in the battle — at least not on the part of the computer.

However, when Kasparov lost to Deep Blue in 1997, emotions did play a role — they even proved to be paramount. While Kasparov is reputedly adept at imposing emotional pressure on his opponents, he is naturally (currently) unable to intimidate a computer; but in an ironic twist of fate during the 1997 match, several unexpected behaviors by Deep Blue caused Kasparov to unwittingly "turn on himself". Following the match, Kasparov admitted to suffering significant emotional distress – even paranoia – as a result of Deep Blue's unexpected behavior [4]. Kasparov's resulting self-imposed distress was partially responsible for his losing the match, and in essence he partially defeated himself. Clearly emotions played a key role in the game, albeit in a very surprising and perhaps totally unexpected manner.

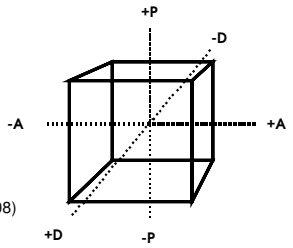
Our second motivation for selecting Chess, then, was to explore the possibility of using negative emotional behaviors (e.g., intimidation) on the part of the computer to achieve similar self-impairing reactions on the part of human opponents. Even if the computer were capable of beating the human without resorting to negative behaviors, doing so might make winning take less time or prove more spectacular. Finally, because chess is a board game rather than an immersive, virtual reality game, we would be limited to low bandwidth in the emotional communication channels [1], resulting in stricter success criteria. In our initial prototype, we have limited our model to vocal behaviors only.

4. Combining Emotional Traits and States

We chose Mehrabian's PAD Temperament Model for the basis of our computational emotion system [5]. This model decomposes emotions into three distinct components — pleasure, arousal, and dominance — each of which is represented numerically in a normalized scale from -1 to +1. Because the three components are (nearly) orthogonal, they can be used as the basis for a 3-dimensional vector space into which various human emotions can be mapped. Figure 1 illustrates this space and lists numeric P-A-D values for several of the more than 240 emotions defined for Mehrabian's model.

The following sample ratings illustrate definitions of various emotion terms when scores on each PAD scale range from -1 to +1:

angry (-.51, .59, .25)
bored (-.65, -.62, -.33)
curious (.22, .62, -.01)
dignified (.55, .22, .61)
elated (.50, .42, .23)
hungry (-.44, .14, -.21)
inhibited (-.54, -.04, -.41),
loved (.87, .54, -.18)
puzzled (-.41, .48, -.33)
sleepy (.20, -.70, -.44)
unconcerned (-.13, -.41, .08)
violent (-.50, .62, .38).



The emotional state "angry" is a highly unpleasant, highly aroused, and moderately dominant emotional state. The "bored" state implies a highly unpleasant, highly unaroused, and moderately submissive state.

Figure 1: The Mehrabian P-A-D Temperament Scale
From: Albert Mehrabian's PAD Scales (see [5]).

Although Mehrabian's PAD system is a largely static system and is used for characterization and classification purposes, we developed a dynamic PAD-inspired model that continually computes the robot's changing emotional state and biases it by the robot's set of long-term, emotional traits. Emotional states are transitory; they can vary substantially and rapidly throughout a game; an emotional trait is a long-term predisposition to be in a particular emotional state.

5. The NTIM Framework

The NTIM framework integrates a traditional chess-playing program with an overarching, emotionally aware behavioral model. It incorporates parameterizable, personality-specific belief models, dynamic emotional states, scalable affective behaviors, and model-based reasoning (see

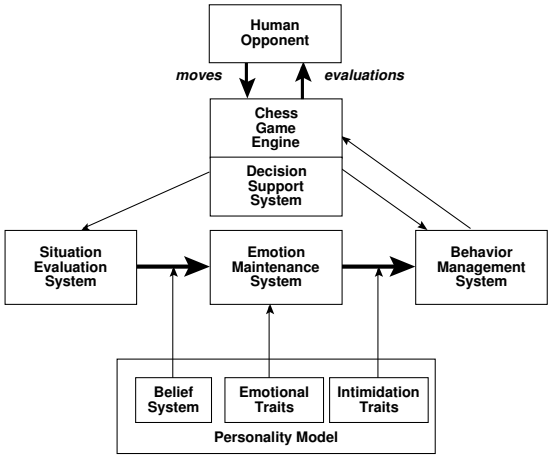


Figure 2: High-Level View of the NTIM Architecture

Figure 2). The NTIM framework contains the following key components:

For the **Chess Playing Engine**, we chose to adopt an "off the shelf" chess program and to tinker with it as little as possible. This greatly simplifies our situational appraisal tasks since the chess engine already computes numerous performance variables and statistics. The **Decision Support System** simply provides mediated access to these variables and game statistics.

The **Situation-Appraisal Manager** performs stimulus evaluation and situational appraisal. Given the computed performance variables and statistics, it "thinks" about the current game situation (board, control, score, etc.) and decides what it "means" to the robot and how the robot "feels" about it -- good or bad. These feelings update the emotional state. The **Emotional-State Manager** maintains the robot's emotional state. Game play and situation-appraisal induce changes in the emotional state, but the ESM uses the robot's temperament to ensure that the robot's temporary states are kept consistent with its personality.

The **Behavior Management System** selects, controls, and expresses the robot's negative, intimidating behaviors. The behavioral choices are purposeful; they are based on the robot's emotional state and beliefs that represents the robot's thoughts on how intimidation best works.

A **Personality Specification Module** can be used to tune the NTIM cognitive, emotional, and behavioral systems. It consists of two separate personality components consisting of "thought amplifiers" — sets of numeric weights used to modify the base signals — and the emotional temperament model — specifications of the robot's emotional traits and their associated strengths. [7]

6. Situational Influence on Emotions

The Situation-Appraisal Manager (SAM) performs low-level, unconscious stimulus evaluations and situational appraisals. It symbolically "thinks about" the current situation and decides what it "means" to the robot. This includes evaluating the current board position, the way the game is proceeding, and the like. These and other performance variables calculated by the chess engine are treated as stimuli by the SAM and converted to situational variables for use in a situational calculus. The situational calculus is used to define potential low-level "thoughts" and "impressions" on the part of the robot; these thoughts affect the robot's emotional state.

Situational appraisal may also involve complex beliefs and memories of game positions and play. We limit these to learned emotional beliefs, and thus we treat them as additional stimuli as well. In this way, though, higher-level cognitive inputs can influence the situational appraisals.

In an attempt to maintain cognitive and neurological plausibility, we model the result of each situational assessment function as a set of signal strengths, or here, as a triple of PAD deltas — potential changes to the system's current emotional state. These objective PAD deltas are weighted by the robot's personality-specific "thought amplifiers" to convert them into subjective PAD deltas. These are then averaged using a weighted sum and passed on to the Emotional State Manager as a single three-component vector for use in updating the robot's current emotional state.

7. The Emotional-State Manager

The Emotional-State Manager (ESM) maintains internal PAD-scale representations of both the robot's static temperament and its dynamic, emotional state. The emotional-state PAD model is biased by the temperament model. We use the following mechanisms to implement this:

- a personality-specific emotional temperament, specified as a set of points and densities in the 3-dimensional PAD space [5]
- an update function that gradually 'walks' through the 3D space, dampened by the temperament model; thus dynamic emotional swings stay relatively consistent with the personality profile
- an emotional state fading mechanism that slowly moves toward the [0,0,0] neutral state when the SAM produces no emotional input

During game play, the SAM reaches various conclusions about how it "feels" about the game; these feelings are represented as PAD deltas and sent to the ESM. The ESM uses these deltas and the current emotional state to compute a new dampened, emotional state. If the SAM does not know what to think about a situation, the emotional state fading mechanism kicks in. All this results in dampened emotional swings, emotional states that fade over time, and emotional states that remain reasonably consistent in terms of the personality's emotional traits.

8. Strategic Use of Negative Emotional Behaviors

We are initially exploring two simple cases in which negative, intimidating behaviors can be exhibited by the robot. First, losing positions may cause it to become angry and choose to rant and rave at the human — to express displeasure, to hopefully upset the human's future play, or whatever. Second, winning situations might make it gleeful and choose to gloat or ridicule the human. The human opponent would likely perceive both behaviors as negative, and either or both might serve to impair his playing ability. However, the first type of behavior might inadvertently please the human, since in this case the

human would already likely perceive himself to be in the superior situation. Observing the robot's pathetic exhortations at losing might at that point serve only to encourage the human and to improve his play. We chose to focus on the second case. Because the human will likely have already perceived that he is in a losing situation, his emotional state is likely to already be negatively oriented at the moment. Thus if the robot chooses to then gloat or ridicule the human, it is in essence "piling on", and the human is more likely to suffer further negative shifts in his emotional state — or so we hope.

Thus we adopt the fundamental assumption that grousing when the human is ahead is more likely to reinforce the human's already positive emotional state and make his play stronger, whereas gloating when the human is behind is more likely to reinforce the human's already negative emotional state and make his play weaker

The NTIM framework contains a model-based method of purposeful intimidation of the human opponent. It controls the type and frequency of exhibited behaviors in an attempt to protect against inappropriately exhibiting negative behavior or exhibiting too much or too little negative behavior. Were it not to do so, then the human may become inured to the negative behavior and recover from its impact. Because we do not yet have empirical results, the NTIM framework's belief model is parameterizable through the personality "thought amplifiers".

9. The Behavior Management System

Once the robot's emotional state has been updated, the Behavior Management System (BMS) decides whether and how to act out in its attempt to intimidate the user. The determination is a function of the robot's emotional state, a belief model of how intimidation best works, and the history of recent intimidation actions. Even though an emotional state may predispose the robot to certain types of behaviors, these can be controlled, over-ridden, or modified; further, the robot can dissimulate.

Our current behavioral repertoire consists only of negative vocalizations (comments, sighs, finger tapping, etc.) recorded as sound files. They are played on cue when the BMS determines that a specific type of negative behavior should be invoked, such as contempt, impatience, disbelief, etc. Each negative behavior type has associated with it one or more negative vocalization files; this helps minimize potential repetitious behavior. We can also record and select from multiple levels of emotional intensity for each negative behavior.

The BMS in the NTIM framework is not a simple "reactive" translator of emotional states to emotional behaviors. It is a self-reflective, cognitive part of the computational emotional process, able to consciously access the underlying primary emotional evaluation (i.e. the ESM), and other data from the system to

actively choose an appropriate behavior within the context of a goal (e.g. intimidating). Therefore, the emotional state from the ESM predisposes the possible set of behaviors, it certainly does not prescribe it, as would be the case in systems implementing a more reactive emotional expression.

10. Summary

We designed the NTIM framework to study the integrated use of emotions in both reactive and conscious, purposeful behaviors. The core of this framework is a computational model of emotions that dynamically integrates emotional states with emotional traits. The framework also incorporates separate appraisal and behavioral modules that are loosely coupled to the emotional maintenance system. To explore the NTIM framework, we are developing a conscious chess-playing robot whose goal is to reason about its own internal emotional state and select appropriate negative behavioral responses that it believes will impair the game playing skills of its human opponent. Our initial prototype is limited to emotional vocal behaviors only; these are expressed by the robot as an integral part of and during the actual playing of the game of chess.

REFERENCES

1. Johnson, Colin and Jones, G.: Design precepts for incorporating affect into intelligent virtual agents, May 2000. (http://www.cs.kent.ac.uk/people/staff/cgj/papers/vr.ps)

2. Marsella, Stacy and Gratch, Jonathan: A step toward irrationality: using emotion to change belief, First Int'l Conf. on Autonomous Agents and Multiagent Systems, Bologna, ACM, July 2002.

3. Plutchik, R.: *The Emotions*, Univ. Press of America, 1991.

4. Chelminski, Rudy: This Time It's Personal: Humankind battles to reclaim the chess-playing championship of the world, *Wired*, Issue 9.10, Oct. 2001.

5. Mehrabian, Albert: Pleasure-Arousal-Dominance: A General Framework for Describing and Measuring Individual Differences in Temperament, *Current Psychology*, Winter, 1996, V14, N4, pp. 261-292.

6. McCauley, Thomas L. and Franklin, Stan: An Architecture for Emotion, AAAI 1998 Fall Symposium, *Emotional and Intelligent: The Tangled Knot of Cognition*, AAAI, 1998.

7. DeGroot, Doug and Broekens, Joost: Integrating Emotional Traits and States in Virtual Humans, in preparation.

Evolutionary Concept Learning with Constraints for Numerical Attributes

Federico Divina Maarten Keijzer Elena Marchiori

Department of Computer Science
Vrije Universiteit
De Boelenlaan 1081a, 1081 HV Amsterdam
{divina,mkeijzer,elena}@cs.vu.nl

Abstract

This paper proposes two alternative methods for dealing with numerical attributes in inductive concept learning systems based on genetic algorithms. The methods use constraints for restricting the range of values of the attributes and novel stochastic operators for modifying the constraints. These operators exploit information on a subset of thresholds on numerical attributes. The methods are embedded into a GA based system for inductive logic programming. Results of experiments on various data sets indicate that the methods provide effective local discretization tools for GA based inductive concept learners.

1 Introduction

Inductive Concept Learning (ICL) [11] constitutes a central topic in Machine Learning. The problem can be formulated in the following manner: given a description language used to express possible hypotheses, a background knowledge, a set of positive examples, and a set of negative examples, one has to find a hypothesis which covers all positive examples and none of the negative ones. The so learned concept can be used to classify previously unseen examples. When hypotheses are expressed in (a fragment of) first order logic, ICL is called Inductive Logic Programming (ILP).

Many learning problems use data containing numerical attributes. Numerical attributes affect the efficiency of learning and the accuracy of the learned theory. In ILP, numerical attributes are either treated as nominal ones, or they are discretized into intervals that will be used as nominal values instead of the continuous values. The discretization can be done during the learning process (*local* discretization), or beforehand (*global* discretization). Discretization methods that employ the class information of the instances are called *supervised*, while if they do not use this information they are called *unsupervised*. A simple discretization is the equal interval width method, where the continuous values are divided into n equal sized bins, where n is a parameter. A better way for discretizing numerical attributes was proposed by Fayyad and Irani [8]. This method uses a recursive entropy minimization algorithm and employs the Minimum Description Length principle in

the stopping criterion. In [10] a variant of the Fayyad and Irani's method is used in an Inductive Logic Programming system. Previous work on GA based learners that explicitly deal with numerical attributes include the following three approaches. In [2], methods using adaptive discrete intervals are used within a GA based system for classification. In [1] a natural coding is used, which transforms the attributes domain (continuous and discrete) in a finite set of natural numbers. In [6], an unsupervised, global method is used to estimate the density of each attribute. This is used to guide genetic operators defined on constraints for numerical attributes. Another approach for local discretization is proposed by Kwedlo and Kretowski, where information on a subset of all thresholds of a numerical attribute is used to determine constraints on numerical attributes in evolved decision rules [9].

In this paper we propose two alternative methods for treating numerical attributes. The methods use the Fayyad and Irani's algorithm and a subset of thresholds on numerical attributes in a way similar to [9]. We embed these methods in a Evolutionary Algorithm (EA) system for ILP [5], and analyze empirically their effectiveness.

The paper is structured in the following way. In section 2 we illustrate with a typical example why in some cases univariate global discretization does not work. In section 3 we describe the methods for dealing with numerical attributes, and in section 4 we describe how these methods are embedded into the ILP system. In section 5 we perform experiments and discuss the results.

2 Weak Point of Univariate Discretization

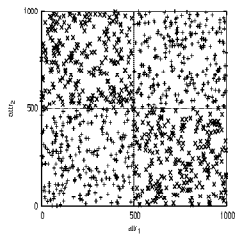


Figure 1: An artificially generated dataset, for which univariate global discretization is unlikely to work.

Univariate discretization methods, e.g Fayyad and Irani's algorithm, are not able to capture the interdependencies among attributes. For this reason globally discretizing numerical attributes with such methods yields the risk of missing the necessary information in order to find the correct solution. A classical example for illustrating this problem is shown in figure 1. In this problem, every example is described by two attributes $attr_1$ and $attr_2$ uniformly distributed on the interval [0,1000]. The examples are equally divided in two classes, with positive examples illustrated in figure 1 by + and negatives by $\times$. The solution to this learning problem is represented by the two lines in figure 1, described by the two rules

$(attr_1 \leq 500) \wedge (attr_2 \leq 500)$ and $(attr_1 > 500) \wedge (attr_2 > 500)$. If the values of the numerical attributes are discretized with the Fayyad and Irani's algorithm, a solution can not be found. In fact, the discretization will result in a unique interval for both attributes. This because the condition $attr_i \leq 500$ or $attr_i > 500$, $1 \leq i \leq 2$, splits the examples in two sets that have the same class distribution. This suggests that in some cases the use of an univariate discretization method could negatively affect the accuracy of the learned theory.

3 Handling Numerical Attributes

We handle numerical attributes by using constraints of the form $l < X \leq u$, where X is a variable relative to a numerical attribute and l, u are thresholds. During the execution of a GA based learner, a constraint for a numerical attribute is generated when that attribute is selected. Constraints are then modified during the evolutionary process by using the operators defined in this section. These operators use information on a set of thresholds on numerical attributes which are computed, like in [7] as follows.

For every numerical attribute A, we first find a set of boundary threshold couples (l_i, u_i), where l_i is the lower bound and u_i is the upper bound. The values l_i and u_i are computed in the following way. The values of A are sorted in increasing order. The values are then divided into n intervals, whose thresholds are represented by (l_i, u_i), $1 \leq i \leq n$. (l_i, u_i)'s are here called Minimum Thresholds (MT) intervals. l_i, u_i are midpoints between successive values of A belonging to examples of different classes. Special cases are $l_1 = -\infty$ and $u_n = +\infty$. An example of MT intervals for an attribute A is presented in figure 2. An interval is considered negative (resp. positive) if its values occur only in negative examples (resp. positive). An interval is mixed if it is relative to just one value, which occurs both in positive and negative examples.

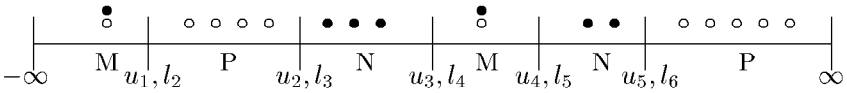


Figure 2: An example of intervals for a numerical attribute A. $l_1 = -\infty$ and $u_6 = \infty$. M denotes a mixed interval, P a positive interval, and N a negative interval. ∘ stands for a value occurring in a positive example, while • for a value occurring in a negative example.

A constraint on A is defined using upper and lower bounds that are chosen among l_i, u_j, with $i \leq j$. The constraint can include one or more MT intervals, with the limitation that a constraint has to begin and end in a positive or mixed interval. For instance, the constraint $l_2 < X \leq u_4$ for A is an allowed constraint, while $l_5 < X \leq u_6$ is not, since its lower bound is the lower bound of a negative MT interval.

We introduce two natural ILP generalization and specialization operators for constraints: *enlarge* and *shrink*. In the following, let us assume that the operators

are called on a constraint C of the form $l_i < X \leq u_j$ relative to a numerical attribute A .

Enlarge This operator called on C returns a constraint C' where one of the bounds is enlarged to the next threshold identifying a MT interval. More precisely the operator works in two steps:

1. Randomly select either l_i or u_j to be changed;
2. (a) If l_i has been chosen, try to modify it to the nearest lower bound l_t, $t < i$ relative to a positive or mixed MT interval. $C' = l_t < X \leq u_j$. If this is not possible perform step (b). If also step (b) is not possible then $C' = C$. Return C'.
   (b) If u_j has been chosen, try to modify it to the nearest upper bound u_t, $t > j$ relative to a positive or mixed MT interval. $C' = l_i < X \leq u_t$. If this is not possible perform step (a). If also step (a) is not possible then $C' = C$. Return C'.

In step 2 enlarging from one side is not possible if there is no positive or mixed interval on that side. For example if C is defined over the intervals depicted in figure 2, and $C = -\infty < X \leq u_2$, then C can not be enlarged in the first point of step 2.

Shrink This operator called on C returns a constraint C' where one of the bounds is shrunk to the next threshold identifying a MT interval. Also this operator acts in two steps:

1. Randomly select either l_i or u_j to be changed;
2. (a) If l_i has been chosen, try to modify it to the nearest lower bound l_t, where $t > i, t \leq j$, relative to a positive or mixed MT interval. $C' = l_t < X \leq u_j$. If it is not possible then perform step (b). If also step (b) is not possible then $C' = C$. Return C'.
   (b) If u_j has been chosen, try to modify it to the nearest upper bound u_t, where $t \geq i, t < j$, relative to a positive or mixed MT interval. $C' = l_i < X \leq u_t$. If it is not possible then perform step (a). If also step (a) is not possible then $C' = C$. Return C'.

4 The Methods

In order to test the effectiveness of the proposed operators, we embed them into the Evolutionary Concept Learner (ECL) system [5] illustrated in figure 3.

For lack of space, we cannot here describe the system in detail. Like many ILP systems, ECL treats numerical attributes as if they were nominal, therefore it can be used as a platform for testing and comparing discretization methods. Selected individuals are mutated and then optimized. Mutation consists of the application of the standard mutation operators of ECL, or in the application of the two operators introduced for numerical attributes. Optimization consists of

```
ALGORITHM ECL
Sel = positive_examples
repeat
    Select partial Background Knowledge
    Population = ∅
    while (not terminate) do
        Adjust examples weights
        Select n chromosomes using Sel
        for each selected chromosome chrm
            Mutate chrm
            Optimize chrm
            Insert chrm in Population
        end for
    end while
    Store Population in Final_Population
    Sel = Sel - { positive examples
        covered by clauses in Population }
until max_iter is reached
Extract final theory from Final_Population
```

Figure 3: The overall learning algorithm ECL

the repeated application of the mutation step until the fitness of the individual increases, or a maximum number of optimization steps has been reached. In the former case the last mutation step applied is retracted. The fitness of an individual is given by the inverse of its accuracy.

In ECL when an individual has to be created a positive example is chosen as a seed. Atoms of the background knowledge describing the seed example are added to the body of the emerging rule. Whenever an atom describing the value of a continuous attribute is added to the body of the rule, also a constraint relative to that attribute is added.

We propose two methods for initializing constraints:

MT In this setting the lower and upper bound of the constraint are those of the MT interval containing the value of the attribute describing the positive example chosen as seed.

FT In this setting the lower and upper bounds are the lower and the upper bounds of the interval containing the value of the attribute found with the Fayyad and Irani's discretization algorithm. In this way a constraint can include more MT intervals. We enforce that the upper and lower bound of the introduced constraint lie inside a positive or mixed MT interval.

As an example of these initializations, suppose that the seed example has value v for the numerical attribute A (see figure 2), and $l_2 < v \leq u_2$. Then the introduced

constraint will be $l_2 < X \leq u_2$ in the MT case, and $l_2 < X \leq u_4$ in the FT case (suppose that (l_2, u_4) is an interval found with the Fayyad and Irani's algorithm).

5 Experiments

We consider four variants of ECL: ECL-FT where numerical attributes are handled using FT initialization and enlarge and shrink operators are used, ECL-MT, where numerical attributes are handled using MT initialization and enlarge and shrink operators are used. Clu-Con, a variant of ECL introduced in [6] where constraints on numerical attributes are obtained by estimating the density distribution of attributes values and variants of enlarge and shrink are used. EntMDL where numerical attributes are discretized with Fayyad and Irani's algorithm and used as nominal values in ECL.

5.1 An Artificially Generated Dataset

We first test the proposed methods on an instance of the problem illustrated in section 2. Our example is made of 50 positive examples and 50 negative examples, each described by two numerical attributes $attr_1$ and $attr_2$ which can assume values in the range $[0, 1000]$. The distribution of examples is similar to the one shown in figure 1. EntMDL can not find a solution for this problem, since both the attributes are discretized in a unique interval. Instead ECL-MT was able to find a solution for this problem, represented by the following two clauses:

$case(X) : -attr_1(X, Y), attr_2(X, Z), (611.56 < Z \leq 976.55), (516.79 < Y \leq 975.36).$
$case(X) : -attr_2(X, Z), attr_1(X, Y), (-\infty < Z \leq 487.58), (21.24 < Y \leq 489.59).$

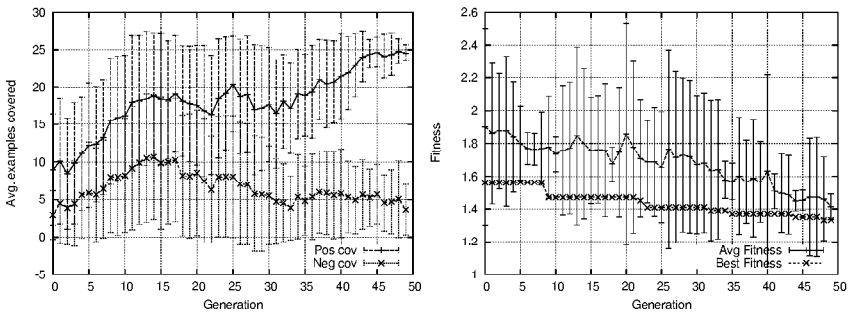


Figure 4: Graphs for a typical run with the MT setting.

These clauses represent a solution of 100% accuracy for the problem instance. The system used a population size set to 100, for 50 generations and with 30 individuals selected per generation. The left graph in figure 4 shows the average number of examples covered by each individual in the population at every iteration, while the right one shows the average and the best fitness per each generation.

ECL-MT initializes constraints to minimal intervals, so that a new generated clause covers one positive example and no negative examples. Constraints can be then enlarged little by little to other MT intervals, and this allows the system to improve, even if slowly, the accuracy of the clause.

ECL-FT was not able to find a solution for this problem, because in ECL-FT the constraints initially introduced in the clauses are initialized to the intervals found with the Fayyad and Irani's algorithm, which are $(-\infty, +\infty)$, and application of enlarge and shrink operators decreases the accuracy of the rule. In the Clu-Con setting the solution found had an accuracy of 98%.

5.2 Benchmark Datasets

In this section we will test the four methods on various benchmark problems. The mutagenesis dataset originates from [4], while others are taken from [3]. We have chosen to experiment on these datasets because they are characterized by a high number of numerical attributes. We used ten-fold cross validation. Each dataset is divided in ten disjoint sets of similar size; one of these sets is used as a test set, and the union of the remaining nine forms the training set. Then the various variant of ECL are run on the training set and the resulting logic program, is tested on new examples using the test set. Three runs with different random seeds are performed on each dataset.

Dataset	ECL-FT	ECL-MT	Clu-Con	EntMDL
Australian	**0.85 (0.03)**	0.83 (0.06)	**0.85 (0.04)**	0.84 (0.05)
Breast	**0.95 (0.02)**	0.94(0.03)	0.94 (0.02)	0.93 (0.02)
Crx	**0.84 (0.05)**	0.82 (0.04)	0.83 (0.05)	0.82 (0.05)
Echocardiogram	0.74 (0.14)	**0.77 (0.14)**	0.65 (0.15)	0.69 (0.19)
German	**0.74 (0.04)**	0.72 (0.04)	0.71 (0.04)	**0.74 (0.04)**
Glass2	0.85 (0.09)	0.75 (0.14)	0.71 (0.07)	**0.86 (0.08)**
Heart	**0.80 (0.07)**	0.73 (0.09)	0.77 (0.07)	0.77 (0.09)
Hepatitis	0.83 (0.07)	**0.84 (0.10)**	0.80 (0.09)	0.83 (0.06)
Ionosphere	**0.89 (0.07)**	0.88 (0.04)	0.78 (0.09)	0.87 (0.06)
Pima-Indians	**0.76 (0.05)**	0.71 (0.05)	0.70 (0.05)	0.68 (0.06)
Mutagenesis	0.88 (0.07)	**0.90 (0.04)**	0.86 (0.10)	0.89 (0.07)

Table 1: Results obtained. The best results obtained by the different settings of ECL are highlighted.

The average accuracy and standard deviation (between brackets), over all three runs, are given in table 1. It can be seen that in most of the cases ECL-FT is more effective than the other methods. ECL-MT is superior only in three cases (Echocardiogram, Hepatitis, Mutagenesis), and the EntMDL is superior only on Glass2. And also in these cases the accuracy obtained by the other methods is close to the accuracy of the solution found by ECL-FT. ECL-FT performed much better then ECL-MT on the Glass2 and the Pima-Indians dataset. This because ECL-FT could initialize constraints using large sub-optimal intervals, and then

slightly adapt them to fit the training set. ECL-MT initialized constraints to small intervals, and then it could not perform enough mutation steps in order to enlarge the intervals in such a way to correctly cover more examples.

The results indicate that local supervised discretization methods (ECL-FT,ECL-MT) are more effective than global supervised ones (EntMDL) and local unsupervised ones (Clu-Con). The efficiency of the algorithms is comparable, with EntMDL being slightly faster than the others. Therefore we can conclude that supervised local discretization using generalization/specialization operators for constraints provides an effective way for dealing with numerical attributes in GA based ICL systems.

References

[1] Jesús S. Aguilar-Ruiz, José C. Riquelme, and Carmelo Del Valle. Improving the evolutionary coding for machine learning tasks. In *European Concefence on Artificial Intelligence, ECAI'02*, pages 173–177, Lyon, France, 2002. IOS Press.

[2] Jaume Bacardit and Josep Maria Garrell. Evolving multiple discretizations with adaptive intervals for a Pittsburgh rule-based learning classifier system. In *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 1818–1831, Chicago, 12-16 July 2003. Springer-Verlag.

[3] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[4] A.K. Debnath, R.L. Lopez de Compadre, G. Debnath, A.J. Schusterman, and C. Hansch. Structure-Activity Relationship of Mutagenic Aromatic and Heteroaromatic Nitro Compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medical Chemistry*, 34(2):786–797, 1991.

[5] F. Divina and E. Marchiori. Evolutionary concept learning. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 343–350, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[6] Federico Divina, Maarten Keijzer, and Elena Marchiori. A method for handling numerical attributes in GA-based inductive concept learners. In *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, pages 898–908, Chicago, 12-16 July 2003. Springer-Verlag.

[7] U. Fayyad and K.B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Mach. Learn.*, 8:87–102, 1992.

[8] U. Fayyad and K.B. Irani. Multi-interval discretization of continuous attributes as pre-processing for classification learning. In *Proceedings of the 13th International Join Conference on Artificial Intelligence*, pages 1022–1027. Morgan Kaufmann Publishers, 1993.

[9] Wojciech Kwedlo and Marek Kretowski. An evolutionary algorithm using multivariate discretization for decision rule induction. In *Principles of Data Mining and Knowledge Discovery*, pages 392–397, 1999.

[10] W. Van Laer, S. Dzeroski, and L. De Raedt. Multi-class problems and discretization in icl. In *Proceedings of the MLnet Familiarization Workshop on Data Mining with Inductive Logic Programming*, pages 53–60, 1996.

[11] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

Intelligent Maintenance Scheduling using an Expert-Driven Fuzzy-Rule Based Object Quality System

Richard van Duijn [a] Jan van den Berg [b] Mark Vreijling [a]

[a] ZooRobotics, Jan Ligthartstraat 1, 1817MR Alkmaar
email: {vanduijn,vreijling}@zoorobotics.com
[b] Erasmus University Rotterdam, P.O.Box 1738, 3000DR Rotterdam
email: jvandenberg@few.eur.nl

Abstract

Optimal maintenance management concerns a highly complex problem requiring intelligent system support. In this paper we propose a generic architecture of a scheduling module for preventive maintenance of a set of objects consisting of three units: (a) an object model, (b) a fuzzy rule-based quality system, (c) a maintenance scheduling unit. Most emphasis is put on the creation of the quality system. We sketch how available expert knowledge can be exploited to construct a fuzzy rule base and a parameterized quality curve and how these two can be combined into one object quality system. The system can be visualized which strongly supports expert-driven validation and testing. Using Mamdani fuzzy reasoning (which implements the 'computing with words'), the system is able to translate state information on a given object as collected through an inspection, into its specific quality curve. The quality curve in its turn determines the moments when maintenance actions are recommendable, desired or even urgent. Having collected this kind of information of all objects to be maintained, the scheduling unit is used to calculate an optimal planning scheme for maintenance execution[1].

1 Introduction

Due to growing numbers of objects to be maintained and increasing demands related to environmental measurements and availability, adequate maintenance execution has become a complex and expensive task. Around 15 years ago it was already recognized that a shift from corrective maintenance planning to a preventative approach can help to reduce costs [5]. However, knowing the optimal moments to execute maintenance activities concerning all objects is not simple: we need all kinds of information from the domain at stake to be able to estimate future objects' states and, next to that, system support for optimal maintenance scheduling and work planning.

[1] This paper is largely based on the content of master thesis [9].

Looking at current Computerized Maintenance Management Software (CMMS), we notice that the possibilities of information storage, updating, and distribution have improved a lot. In addition to this, CMMS often has modules available for doing some risk management and for streamlining processes related to the execution of maintenance. What is still missing however are multi-channel access to the software and, even more importantly, efficient and effective maintenance planning modules.

The goal of this paper to sketch an appropriate architecture of the scheduling module of future CMMS systems where the maintenance approach is preventive. Such a planning module is supposed to consist of three units: (a) an object model, (b) a fuzzy rule-based quality system, (c) a maintenance scheduling module. Most emphasis is put on the creation of the fuzzy-rule based quality model. The rest of this article is structured as follows. In section 2, we introduce the fundamental building blocks of a software module for intelligent maintenance planning. Next in section 3, we elaborate the fuzzy rule-based object quality model by giving details on its creation, visualization and validation capabilities. We also dwell upon on things to do next (section 4). We finalize by giving some conclusions.

2 Generic Architecture

In order to apply intelligent maintenance planning in a wide range of domains, we developed a generic multitier architecture of the necessary software modules being a presentation tier, a database tier and a business logic tier. For the purpose of this article, the above-mentioned units from the business logic tier are of importance, i.e. the object model, the quality system, and the maintenance scheduling module.

2.1 Object Model

Setting up a generic object structure is fundamental to the implementation of intelligent maintenance software. We assume having available the set of *elements* to be maintained (as a working example we take a set of pylons used for supporting high-tension electricity cables). Then, any part of an element that during inspection or maintenance activity is considered as a whole, will be termed an *object* (for the working example, we can choose to define a traverse and a pylon foot as being objects). Defining objects this way enables us to link a set of maintenance actions to each object. In the software, these actions are represented by corresponding 'methods'. In practice, objects can be determined based on discussions with domain experts: all elements to be maintained should in some way 'logically' be decomposed in objects.

2.2 Fuzzy Rule-Based Object Quality Curves

Having defined objects and the corresponding actions, we want to make an optimal planning. To do so we need to know the moments of all objects when maintenance can or should be performed. To be able to estimate the moments of next maintenance, we assume that it is possible to

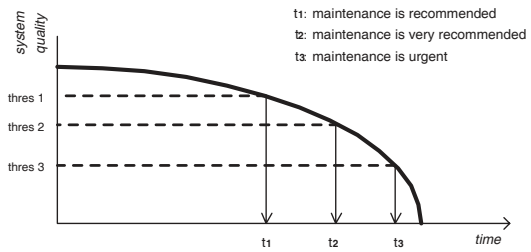


Figure 1: Quality curve with threshold values determining the moments and the type of possible maintenance actions.

1. assess the *quality curve* of all objects based on collected inspection data;
2. estimate the *moments of next maintenance actions* given the quality curve;
3. determine an optimal *maintenance scheduling* scheme.

The way step 2 and 3 of these activities can be performed will be sketched in section 2.3. In this paper, our focal point is on the first step, i.e., the determination of quality curve for all objects (section 3).

2.3 Maintenance Scheduling

Knowing (an approximation of) the quality curve of each object, experts can usually estimate the time to next maintenance or, more generally, the time to the next moments that maintenance is advisable, should be done or is really necessary. To explain we first observe that it is natural to assume that the object quality curve is a monotonically decreasing function of time. Then the estimation of the next maintenance moments can be based on a set of *threshold values* 'thres 1', 'thres 2', ... in the quality curve. The determination of the threshold values is done by domain experts. Each threshold 'thres i' corresponds to a time moment t_i that maintenance is 'recommended', 'very recommended', 'urgent', and so on: see Figure 1. Here the following rule of thumb holds: the longer the postponement, the more extensive the maintenance work.

In order to set up an optimal scheduling of all maintenance activities, one may start to apply a *single component strategy* where the (best) possible action(s) for each individual object are determined. Here we should reckon with several constraints, for example related to the sequence of maintenance actions: it is known from practice that a series of light maintenance activities should be alternated from time to time with the execution of a severe maintenance action. Next, it is clear that grouping of actions for different objects may result in decreasing costs, for example in case objects lie physically close together or in case several objects require the same type of maintenance. It is the goal of the *multiple component*

strategy to synchronize maintenance actions such that costs are minimized. We like to observe that this minimization problem is also constrained since the number of actions that can be executed at the same time is limited. It is also possible that certain actions can not be executed simultaneously due to practical or budgetary reasons. So in practice, *budget fitting* also influences 'optimal maintenance scheduling'. For an overview on maintenance scheduling, we refer to [1].

3 Finding object quality curves

In this section, we elaborate the activities to fix the fuzzy rule-based quality curves as introduced in subsection 2.2. To get proof of concept, we performed a case study in the area of coating systems[2]. The elements to be maintained are the aforementioned pylons used for high-tension electricity transportation. The objects (i.e., the smallest parts that need maintenance as a whole: see section 2.2) are the traverse, middle-part, and foot of each pylon. No relevant data set appeared to be available but instead, a lot of expert knowledge was at our disposal based on years of experience in all kinds of maintenance activities. After various discussions with the experts it became clear that they were able to offer a lot of information about the relationship between a set of quality-related feature values and the 'overall quality' of an object. Inspections can be executed to assess the quality-related feature values. As explained in section 2.3, it appeared necessary to determine an quality curve for each object. So, in some way these things had to be combined.

Let us somewhat formalize the problem. The goal is to find quality curves $Q = Q_o(t)$ representing the quality Q of object o as a function of time t. The collected state information of each object o is given by feature vector $\mathbf{f}_o = (f_{o1}, f_{o2}, \ldots, f_{oN})$ of N feature vector values f_{oi}. The experts are supposed to be able to link (in linguistic terms) state information about an object with a corresponding quality parameter q_o of it. Based on the value of q_o, we want to determine the corresponding quality curve $Q = Q_o(t)$. Here the following principles holds: the higher the value of q_o, the wider the quality curve and hence (see again Figure 1), the longer the time to next maintenance. So our problem can be redefined as finding the following two mappings:

$$\mathbf{f}_o \rightarrow q_o \rightarrow Q_o(t). \tag{1}$$

3.1 Assessing the quality of objects

As mentioned above, q_o-values need to be fixed from a set of feature values f_{oi}. Examples of the feature variables in this domain are the *macro-environment* (having standardized ISO-classification [6] with well-defined classes $C_1, C_2, \ldots, C_5$) and the *coating system* applied. To implement the mapping, we proposed to apply fuzzy system theory. To do so, we asked the experts to express their knowledge in fuzzy

[2]For confidentiality reasons, we were obliged to make anonymous a lot of practical details of the case study.

rules $R_q, (q = 1, 2, \ldots, Q)$ like

$$\text{If } \text{`}environment \text{ is } C_3\text{'} \text{ and } \text{`}conservation \text{ is light' and } \text{`}f_{o3} \text{ is complex' and } \ldots \text{ then } \text{`}q_0 \text{ is medium'}. \quad (2)$$

Here, the terms 'C_3', 'light', 'complex', 'medium' correspond to fuzzy membership functions which have been defined during discussions with the experts. This work included long conversations on the precise forms of the various fuzzy sets and their mutual overlap. For reasons of simplicity, it was decided to use trapezoidal membership functions. By applying a fuzzy reasoning scheme with Mamdani's fuzzy implication and the max-min composition [4, 2], the fuzzy output value of q_o is calculated. Next, by applying centroid of area defuzzification [2], the crisp output value of q_o is estimated for each combination of the (fuzzy) input values. In this way, the idea of 'computing with words' is realized. It is important to know here that in our case q_o is expressed in years and lies in the interval $[0, 20]$.

The first fuzzy rule base developed contained $5 \times 4^4 = 320$ rules caused by the fact that the first feature variable had five possible values, and the other four variables four values. A close inspection of the rule base revealed that many rules could be removed simply because they contained a lot of so-called 'don't cares'. A first approximation left us with 100 rules, which meant a reduction of over 200 rules. More precise pruning of the rule base resulted in a remaining set of only 43 rules. Due to this pruning steps, the transparency of the fuzzy rule has been greatly enlarged. For example, great insight in the importance of various parameters has been gained. Of course the correctness of fuzzy rules had still to be proven. We will come back to that issue in subsection 3.3.

3.2 Determination of quality curves

Knowing the q_o-values from the output of the fuzzy system, we should be able to calculate the quality curve of each object. To do so, we need to know a mathematical expression of the general shape of the quality curve. There happened to be done some initial research on this problem based on the fitting of data from one specific case. We decided to use these results, i.e., we decided to use the following quality curve function (internal correspondence):

$$Q_o(t) = 1 - a\, t^{b-p_o}, \quad \text{with } \; Q_o(t) \in [0, 1]. \quad (3)$$

Here, $Q_o(t)$ represents the (relative) quality Q (on a scale from 0 to 1) of object o as a function of time t while a, b, and p_o are positive parameters. It should be clear that, given values for p_o and b, the value of a determines whether the quality curve decreases faster or slower to zero. The experts offered the information that after having finished a conservation of an object, the quality of an object in the beginning decreases quite slowly. The afore-mentioned initial curve fitting research yielded indeed a small value for a, namely $a = 0.00216$. We decided to use this value and to make it the same for the quality curve of all objects. The fitting research also yielded a value for the parameter b, namely $b = 4.16$ and we also decided to use this value for all objects. As a consequence of these choices, the

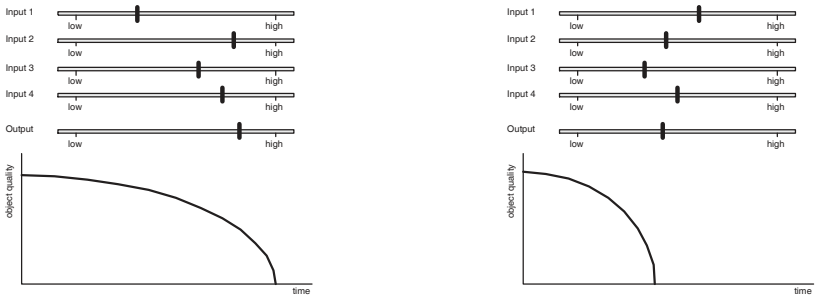


Figure 2: Visualization of fuzzy system showing how, in two cases (left and right), 4 input slide positions are translated into the corresponding position of the output slide and corresponding quality curve.

remaining problem is to fix the parameter p_o for each object. The value of this parameter has been made dependent on the value of the fuzzy system output value variable q_o and has been chosen based on discussions with the experts in the field yielding $p_0 = q_o/10$. The resulting curve used is therefore

$$Q_o(t) = \max(0, 1 - 0.00216\, t^{4.16-q_o/10}). \tag{4}$$

Note that since $q_o \in [0, 20]$ (see the previous section), the exponent of t will be in the interval $[2.16, 4.16]$. In this way, we have constructed a first prototype of a fuzzy rule-based system that implements the desired mappings of (1).

3.3 First inspections of correct working

The working of our prototype as found in the previous step has to be validated. Up till now, the only knowledge available is that of experts, so again, we have to talk with them. To facilitate the discussions, we have made a visualization where the input feature vectors f_{oi} and the output parameter p_o are represented as a slide on a bar. By moving the position of the input slides, the position of the output slide is adapted automatically based on the above-described fuzzy rule-based system. At the same time, the corresponding object quality curve is adapted and shown to the user of the system. In Figure 2 we have shown the idea where in two cases, the positions of four input slides are translated in two output positions and corresponding quality curve [3].

The first impressions of our implementation are positive. By gradually changing one or more slides of the input bars, we sometimes note that the output bar does not changes position nor the quality curve. However, in many cases the output bar and quality curve changes too. The reason for this behavior can easily be explained. The first phenomenon of no output chance occurs if only one rule

[3] Maybe needless to say but note how - given the quality curve of an object, given the threshold values 'thres i', and using the mapping as visualized in Figure 1 - the several moments of next maintenance can be determined for that object.

'fires' [2], the second one if several rules fire at the same time because of overlapping membership functions in the input space.

4 Next Steps

Our first next step is to submit our visualized fuzzy system to a group of domain experts in order to validate the system. To do so, a validation set will be constructed with a group of representative examples. In addition to that, experts will be enabled to play with the system according to their own insights. If in certain cases the output does not correspond to what experts expect, we will look at the underlying fuzzy rules that fire and discuss which rule is possibly wrong. By adapting rules, we shall try to optimize the system as much as possible. If needed, we can also change the shape of the membership functions or the quality curve but this is seen as a second best approach. Next to this, we shall try to compose a separate group of domain experts to test the system after having completed the validation procedure just described. Hopefully, the latter group will be quite satisfied concerning the performance of our fuzzy rule-based system.

We shall also start to collect data concerning the input and output variables of the system. This step will enable data-driven fine tuning of our system in times to come, e.g. by introducing rule weights which can be calibrated by application of a learning procedure [2] . Having available data sets, another future approach is to calculate the quality curves of all objects directly using Tagaki-Sugeno modelling [7]. The corresponding fuzzy rules can be formulated as

$$\begin{aligned} &\text{If } `f_{o1} \text{ is small' and } `f_{o2} \text{ is light' and} \\ &`f_{o3} \text{ is complex' and } \ldots \text{ then } Q_o(t) = 1 - at^{b-p_o}. \end{aligned} \tag{5}$$

A last idea is to incorporate probabilistic uncertainty as present in quality curve in our modelling. To do so, we can make use of the the approach called 'probabilistic fuzzy modelling' [8, 3]. By all these activities, we expect to be able to arrive at a robust methodology for modelling quality curves and to master the corresponding technology. This technology can next be tried in other domains.

5 Conclusions

In this paper, we sketched the a general outline of a model for maintenance planning hereby focussing on the modelling of fuzzy rule-based quality curves. We described the implementation of a prototype based on a case study in the coating domain. Since data set where unavailable, we relied on expert knowledge. Expert knowledge is usually not precise and for this reason, we decided to use fuzzy system theory. The resulting first prototype has been visualized for reasons of better understanding of its working and for validation and testing purposes. The first validation results are quite promising and it is hypothesized that the proposed type of intelligent maintenance planning can be used in many other domains.

We also dwelled upon possible improvements of the system. (Adaptive) fine-tuning of the fuzzy rule base can be implemented provided adequate data sets

have been collected. Next to that, it can be advantageous the redesign the fuzzy system using Tagaki-Sugeno fuzzy systems and probabilistic fuzzy modelling. Future experimentation is needed to make our approach really robust and ready to use other domains.

References

[1] R. Dekker, F.A. Van der Duyn Schouten, and R.E. Wildeman. A review of multicomponent maintenance models with economic dependence. *Mathematical Methods of Operations Research*, 45:411–435, 1997.

[2] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, editors. *Neuro-Fuzzy and Soft Computing*. Prentice Hall, New Jersey, 1997.

[3] Uzay Kaymak, Willem Max van den Bergh, and Jan van den Berg. A fuzzy additive reasoning scheme for probabilistic mamdani fuzzy systems. In *Proceedings of the 12-th IEEE International Conference on Fuzzy Systems*, St. Louis, USA, May 2003.

[4] E. H. Mamdani. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[5] J.G. Monk. *Operations Management, Theory and Problems*. McGraw Hill, New York, 1987.

[6] International Standards Organisation. *Paints and Varnishes – Corrosion Protection of Steel Structures by Protective Paint Systems*. 1997.

[7] Tomohiro Takagi and Michio Sugeno. Fuzzy identification of systems and its applications to modelling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1):116–132, January/February 1985.

[8] Willem Max van den Bergh, Uzay Kaymak, and Jan van den Berg. On the data-driven design of tagaki-sugeno probabilistic fuzzy systems. In *Proceedings of the EUNITE Conference 2002*, Portugal, September 2002.

[9] R. van Duijn. Dynaform, ontwikkeling van een intelligent onderhoudsmanagement systeem, February 2003. *Master Thesis*.

Genetic Programming for Data Classification: Refining the Search Space

J. Eggermont J.N. Kok W.A. Kosters

Leiden Institute of Advanced Computer Science
Universiteit Leiden
P.O. Box 9512, 2300 RA Leiden
The Netherlands

Abstract

When using Genetic Programming to evolve decision trees for data classification the search space sizes tend to become extremely large and sometimes theoretically infinite in size, depending on the representation. In this paper we present a method derived from the well-known decision tree construction algorithm C4.5 to refine and thereby reduce the search space sizes for our decision tree evolvers. We will show that using this refinement method we can significantly improve classification performance.

1 Introduction

Data classification is a particularly difficult problem for evolutionary algorithms mainly because of two reasons. First of all evolutionary algorithms for data classification have to compete with deterministic algorithms like C4.5 [8] which are faster since they only have to make a small number of passes through the training-set. Secondly, unlike machine learning and in particular neural network approaches, evolutionary algorithms search for a classifier (usually decision trees) while algorithms like C4.5 and Back-Propagation construct or build a classifier based on heuristics, rules or mathematical models. The size of the spaces in which the evolutionary algorithms are searching is usually very large and sometimes even infinite (at least theoretically). Unlike other optimization problems, there is no representation for all possible solutions (decision trees) but rather the final solution quality is partially dependent on the chosen representation, the so-called language bias. Thus instead of searching in the space of all possible classifiers we can only search in a restricted space. However, this does not mean that this search space is by any means small as we will show for different data sets. In this paper we use tree-based Genetic Programming (GP) [1, 7] to evolve decision trees for binary classification problems.

The outline of this paper is as follows. We start by describing the decision tree format that we will be using. Next we show how to calculate the search space size given a set of possible internal and external nodes. We then define three decision tree representations by specifying the set of possible internal and external nodes that can occur in our trees. In Section 4 we describe our experiments followed by

the results. Finally, in the last section we draw some conclusions and look ahead towards future research.

2 Full Atomic Representations

In this paper we use *full atomic* representations. Unlike the *atomic* representation used in [3] which only uses atoms in its terminal set and Boolean operators as internal nodes, a *full atomic* representation also uses atoms in the internal nodes. Each atom is syntactically a predicate of the form (*attribute operator value*(*s*)), where *operator* is a function returning a Boolean value (e.g., $<$, $>$ or $=$). In the leaf nodes we have *class assignment* atoms of the form (*class* $:=$ C), where C is a category selected from the domain of the attribute to be predicted. A small example tree can be seen in Figure 1. A *full atomic* tree classifies an instance I by traversing the tree from root to leaf node. In each non-leaf node an atom is evaluated. If the result is true the right branch is traversed, else the left branch is taken. This is done for all internal nodes until a leaf node containing a *class assignment* node is reached, resulting in the classification of the instance. The advantage of this representation is two-fold. First of all unlike an *atomic* decision tree, we do not have to traverse the entire tree to classify a data record but only a single path from the root of the tree to a leaf (*class assignment*) node. This can potentially save a lot of time. Secondly, the *atomic* representation was limited to binary classifications while a *full atomic* representation can potentially be used for any n-ary classification problem.

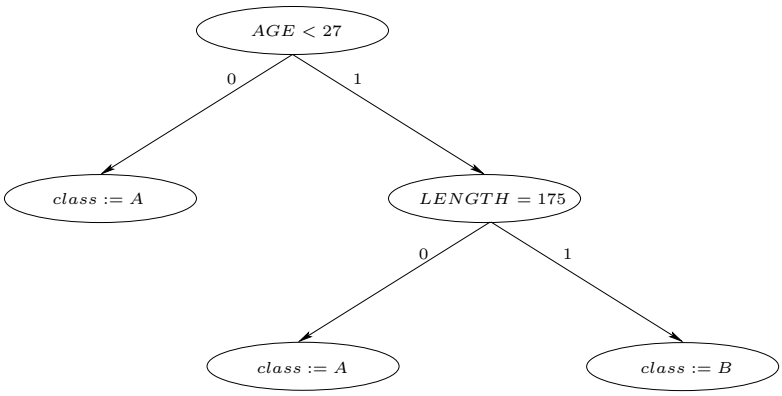


Figure 1: An example of a *full atomic* tree

2.1 Calculating the size of the search space

Since every decision tree using a *full atomic* representation is also a full binary tree we can calculate the size of the search space for each specific *full atomic* representation and data set. In order to calculate the size of the search space of

GP algorithms using a *full atomic* representation for a given data set we use some formulae from discrete mathematics.

Let N be the number of tree nodes. The total number of binary trees with N nodes b_N can be calculated using the Catalan number

$$b_N = \frac{1}{N+1}\binom{2N}{N}. \tag{1}$$

Let n be the number of internal tree nodes. The total number of tree nodes N in a full binary tree with n internal tree nodes is:

$$N = 2n + 1. \tag{2}$$

We can now combine these two equations into the following lemma:

Lemma 2.1 *The total number of full binary trees with* $2n + 1$ *nodes is* $\frac{1}{n+1}\binom{2n}{n}$.

Proof Let B be a tree with n nodes. In order to transform this tree into a full binary tree with $2n + 1$ nodes we need to add $n + 1$ nodes. This can only be done in one way. □

Since in a *full atomic* tree the contents of a node is dependent on the set of internal nodes and the set of external nodes we can define the total number of *full atomic* trees with a maximum tree size of N nodes, a set of internal nodes I and a set of terminal nodes T as follows.

Lemma 2.2 *The total number of full atomic trees with at most* N *nodes (*N *odd), a set of internal nodes* I *and a set of terminal nodes* T *is*

$$\sum_{n=1}^{\frac{N-1}{2}} \frac{1}{n+1}\binom{2n}{n} \times |I|^n \times |T|^{n+1}$$

Note that — contrary to decision trees in algorithms such as C4.5 — the same atom can occur more than once in a path from root to leaf node.

We will use these formulae later to calculate the sizes of the search spaces in our experiments.

2.2 A Simple Representation

By using a *full atomic* representation we have defined the basic shape of our decision trees. We can define the precise decision tree representation by specifying which atoms are to be used. In this section we will specify a simple, but powerful, decision tree representation that uses different types of atoms based on the data type of an atom's *attribute*. For non-numerical attributes we use atoms of the form ($variable_i = value$) for each possible *attribute-value* combination found in the data set. For numerical attributes we define the less-than operator ($<$). Again we use atoms for each possible *attribute-value* combination found in the data set. The idea

in this approach is that the GP algorithm will be able to decide the best *value* at a given point in a tree. This *simple* representation is similar to the representation used by Rouwhorst and Engelbrecht [9].

2.3 Refined Representations

One of the drawbacks of the *simple full atomic* representation is the fact that it creates an internal node for each combination of a numerical valued attribute and value that occurs in the data set. In order to reduce the huge search space size that results from the large number of possible internal nodes we will borrow some ideas from C4.5. Decision tree constructing algorithms like C4.5 tend to be greedy and non-backtracking in nature since finding the smallest decision tree consistent with a specific training-set is NP-complete [5]. In the case of C4.5 and its predecessor ID3 the criteria used were *gain_ratio* and *gain* [8]. These criteria measure the amount of entropy based information gained by splitting a data set on some test. Just like C4.5, our *refined representations* select a single threshold value for splitting a data set on a continuous valued attribute. In effect, from all of the possible internal nodes created by the *simple* representation only one is selected for each numerical valued attribute. Thus for every numerical valued attribute A_i in a data set T we will calculate the *gain* or *gain_ratio* measure for every possible threshold value v_j from the domain D_i of A_i. We will then select for every attribute A_i a threshold value v_t such that $gain(\_ratio)(A_i < v_t) \geq gain(\_ratio)(A_i < v_j)$ *for all* $v_j \in D_i$. We do not expect to see much difference between using the *gain* and *gain_ratio* criteria as we only use them to split the domain of numerical valued attributes into two, but we will try them both to see if there is a significant difference between the two criteria.

2.4 Example

Let us examine the example data set T in Table 1.

Table 1: An example data set

A	**B**	**class**
1	a	yes
2	b	yes
3	c	no
4	d	no

In the case of the *simple* representation we get the following atoms:

- Since attribute **A** has four possible values {1,2,3,4} and is numerical valued we use the $<$ operator: $(\mathbf{A} < 1)$, $(\mathbf{A} < 2)$, $(\mathbf{A} < 3)$ and $(\mathbf{A} < 4)$.
- Attribute **B** is non-numerical and thus we use the $=$ operator: $(\mathbf{B} = a)$, $(\mathbf{B} = b)$, $(\mathbf{B} = c)$ and $(\mathbf{B} = d)$.

- Finally for the target class we have two terminal nodes: ($class := yes$) and ($class := no$).

In the case of the *refined* representations we would have the same atoms for attribute **B** and the target classes. Since attribute **A** is numerical valued we will have to determine a threshold value. Using either the *gain* or *gain_ratio* criteria, ($\mathbf{A} < 3$) would be chosen as the possible internal node as it results in the highest *gain*, and *gain_ratio*.

In the case of the *simple* representation we can calculate the size of the search space to be approximately 3×10^{53}, with our default maximum tree size of 63 nodes. In the case of a *refined* representation we have 5 possible internal nodes and 2 possible class assignment nodes. Given the same maximum tree size of 63 nodes and using Lemma 2.2 this would imply a search space size of approximately 1×10^{47}.

3 Multi-layered Fitness

Although we will compare our *full atomic* GP algorithms to other data classification algorithms based on their classification performance, there is a second objective for our *full atomic* GPs which is also very important: understandability. The *simple* and *refined* representations introduced in the previous section are similar to the decision trees constructed by C4.5 and quite easy for humans to understand. However, even the most understandable decision tree representation can result in incomprehensible trees if the trees become too large. In order to keep the size of our decision trees limited we use two methods. The first method is a built in system which prunes decision trees that have more than a pre-determined number of nodes. The second method is the use of a *multi-layered* fitness.

A multi-layered fitness is a fitness which consists of several fitness measures or objectives which are ranked according to their importance. In the case of our *simple* representation we use a multi-layered fitness consisting of two fitness measures that we want to minimize. The primary, and most important, fitness measure is the misclassification percentage. The secondary fitness measure is the number of tree nodes. When the fitness of two individuals is to be compared we first look at the primary fitness. If both individuals have the same misclassification percentage we compare the secondary fitness measures.

4 Experiments and Results

We will compare our *full atomic* GP representations to C4.5 using several data sets from the UCI machine learning data set repository [2]. An overview of the data sets can be seen in Table 2. More details and comparisons to other evolutionary and non-evolutionary algorithms can be found in the forthcoming PhD thesis of the first author.

Each algorithm is evaluated using n-fold cross-validation and the performance is the average misclassification error over n folds. In n-fold cross-validation the

Table 2: An overview of the data sets used in the experiments

data set	**records**	**attributes**	**classes**
Australian credit (statlog)	690	14	2
German credit (statlog)	1000	23	2
Pima Indians diabetes	768	8	2
Heart disease (statlog)	270	13	2
Ionosphere	351	34	2

total data set is divided into n parts. Each part is chosen once as the test set while the other $n-1$ parts form the training set. In all our experiments we used a 10-fold cross-validation.

We will mention the results of C4.5 as reported by Freund and Shapire [4], in order to compare our results to a non-evolutionary decision tree algorithm.

A single GP implementation was used for both *simple* and *refined* representations. It was programmed using the *Evolving Objects* library (EOlib) [6]. EOlib is an Open Source C++ library for all forms of evolutionary computation and is available from `http://eodev.sourceforge.net`.

In our GP system we use the standard GP mutation and recombination operators for trees. The mutation operator replaces a subtree with a randomly created subtree and the crossover operator exchanges subtrees between two individuals. Both the mutation rate and crossover rate are set to 0.9. The population was initialized using the ramped half-and-half initialization [7, 1] method to create a combination of full and non-full trees with a maximum tree depth of 6. We used a generational model (comma strategy) with population size of 100, an offspring size of 200 and a maximum of 99 generations. Parents were chosen by using (5-)tournament selection. We did not use elitism as the best individual was stored outside the population. Each newly created individual, whether through initialization or recombination, was automatically pruned to a maximum number of 63 nodes.

Table 3: The approximate search space sizes for our algorithms

data set	*simple* GP	*refined* GP
Australian Credit	4×10^{120}	1×10^{61}
German Credit	7×10^{100}	5×10^{67}
Pima Indians Diabetes	3×10^{121}	3×10^{53}
Heart Disease	4×10^{105}	1×10^{60}
Ionosphere Disease	5×10^{146}	9×10^{72}

The search space sizes for our algorithms as calculated using Lemma 2.2 are displayed in Table 3. As can be seen there is a huge difference in search space sizes as the sizes for our *simple* GP algorithm are sometimes more than squared the size

for our *refined* GP algorithms.

Table 4: The average misclassification rates (in %)

		Simple GP	*Refined* GP		*C4.5*
data set			*gain*	*gain_ratio*	
Australian Credit *	average	22.0	**14.3**	15.4	15.9
	s.d.	3.0	0.5	0.4	
	best	17.0	13.6	14.9	
	worst	25.7	15.5	16.1	
German Credit *	average	**27.1**	28.0	28.6	27.2
	s.d.	0.7	0.7	0.5	
	best	26.1	26.9	27.4	
	worst	28.2	28.8	29.1	
Pima Indians Diabetes	average	**26.3**	26.8	27.6	28.4
	s.d.	1.1	0.5	0.0	
	best	24.3	26.0	27.6	
	worst	28.5	27.7	27.6	
Heart Disease *	average	25.2	**19.8**	20.2	22.2
	s.d.	2.3	2.0	1.1	
	best	22.6	15.9	18.1	
	worst	31.1	23.0	21.5	
Ionosphere Disease	average	12.4	**7.6**	7.6	8.9
	s.d.	1.8	0.7	0.9	
	best	8.0	6.5	6.3	
	worst	14.3	8.5	9.1	

We performed 10 independent runs for our two GP algorithms to obtain the results presented in Table 4. When available from the literature the results of C4.5 are reported. For three data sets (Australian credit, Heart disease and German credit) no results were reported for C4.5. In those three instances, marked with a $*$, we applied C4.5 to the data set ourselves. The best results for each data set are printed in bold.

When we look at the results in Table 4 we see that on three of the five data sets our *refined* GP algorithm using the *gain* criterion performs a lot better than our *simple* GP algorithm and C4.5. If we look at the other two data set sets (German Credit and Pima Indians Diabetes), we see these are also the data sets on which our *simple* GP performs better than or close to C4.5. It is at the moment unclear why on these two data sets the results differ from the other three data sets. If we compare the two *refined* GP algorithms we see that using the *gain* criterion results in better classification results on all five data sets. This is somewhat unexpected as C4.5, which uses the *gain_ratio* criterion, is generally considered to be superior to ID3 which uses the *gain* criterion.

Due to the high number of parameters involved it is difficult to draw hard conclusions. *Refined* GP seems to be the overall winner: it is either much better or only slightly worse than the other methods.

5 Conclusions and Future Research

In this paper we have shown how the *gain* and *gain_ratio* criteria from ID3 and C4.5 can be used to refine the search space sizes for GP algorithms. When successful the refinement (and reduction) of the search space helps our GP algorithms to find better (local) optima. Unfortunately, the *gain* and *gain_ratio* criteria do not always help us determine the optimal threshold values.

At the moment our *refined* GP algorithms only select a single threshold value for each numerical valued attribute and thereby split the domain of such an attribute into two. However, the *gain* and *gain_ratio* criteria can also be used to split the domain of numerical valued attribute into n parts (using $n-1$ threshold values). Hopefully extending our *refined* GP algorithms in this way will improve classification performance on all data sets. The *gain_ratio* criterion should be especially useful as it was designed to find a balance between information gained by splitting a data set into a large number of data subsets and limiting the number of subsets.

References

[1] W. Banzhaf, P. Nordin, R.E. Keller, and F.D. Francone. *Genetic Programming: An Introduction*. Morgan Kaufmann, 1998.

[2] C.L. Blake and C.J. Merz. *UCI Repository of machine learning databases*. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[3] J. Eggermont, A.E. Eiben, and J.I. van Hemert. Adapting the fitness function in GP for data mining. In R. Poli, P. Nordin, W.B. Langdon, and T.C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'99*, volume 1598 of *LNCS*, pages 195–204. Springer-Verlag, 1999.

[4] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996.

[5] L. Hyafil and R.L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5:15–17, 1976.

[6] M. Keijzer, J. J. Merelo, G. Romero, and M. Schoenauer. Evolving objects: a general purpose evolutionary computation library. In P. Collet et al., editor, *Proceedings of Evolution Artificielle'01*, LNCS. Springer Verlag, 2001. To appear.

[7] J.R. Koza. *Genetic Programming*. MIT Press, 1992.

[8] J.R. Quinlan. *Programs for machine learning*. Morgan Kaufmann, 1993.

[9] S. E. Rouwhorst and A. P. Engelbrecht. Searching the forest: Using decision trees as building blocks for evolutionary search in classification databases. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 633–638. IEEE Service Center, 2000.

Design By Contract
Deontic Design Language for Component-Based Systems

Christophe Garion[a] Leendert van der Torre[b]

[a] SUPAERO, Toulouse, France
[b] CWI, Amsterdam, the Netherlands

Abstract

Design by contract is a well known theory that views software construction as based on contracts between clients (callers) and suppliers (routines), relying on mutual obligations and benefits made explicit by assertions. However, there is a gap between this theory and software engineering concepts and tools. For example, dealing with contract violations is realized by exception handlers, whereas it has been observed in the area of deontic logic in computer science that violations and exceptions are distinct concepts that should not be confused. To bridge this gap, we propose a software design language based on temporal deontic logic. We also discuss the relation between the normative stance toward systems implicit in the design by contract approach and the intentional or BDI stance popular in agent theory.

1 Introduction

Design by contract [10, 11, 12] is a well known software design methodology that views software construction as based on contracts between clients (callers) and suppliers (routines), relying on mutual obligations and benefits made explicit by assertions. It has been developed in the context of object oriented programming, it is the basis of the programming language Eiffel, and it is well suited to design component-based and agent systems. However, there is still a gap between this methodology and formal tools supporting it. For example, dealing with contract violations is realized by exception handlers, whereas it is well known in the area of deontic logic in computer science [13, 18] that violations and exceptions are distinct concepts that should not be confused. Formal tool support for design by contract is therefore a promising new application of deontic logic in computer science [19]. In this paper we study how deontic logic can be used as a design language to support design by contract. We address the following three research questions.

1. Which kind of deontic logic can be used as a design language to support design by contract?
2. What kind of properties can be formalized by such a design logic?
3. How does this approach based on deontic logic compare to the BDI approach, dominant in agent based software engineering?

The motivation of our work is the formal support for agent based systems. Recently several agent languages and architectures have been proposed which are based on obligations and other normative concepts instead (or in addition to) knowledge and goals, or beliefs, desires and intentions. In artificial intelligence the best known of these normative approaches is probably the IMPACT system developed by Subrahmanian and collegues [6]. In this approach, wrappers built around legacy systems are based on obligations. However, we are interested in particular in component based agent systems such as the BOID architecture [3].

The layout of this paper is as follows. In Section 2 we discuss design by contract. In Section 3 we discuss contract violations. In section 4 we compare this approach based on deontic logic to the BDI approach.

2 Design by contract

We explain design by contract by an example program in the Eiffel programming language. The explanation of design by contract as well as the example have been taken from [9]. For further details on design by contract, see [10, 11, 12].

Design By Contract views software construction as based on contracts between clients (callers) and suppliers (routines), relying on mutual obligations and benefits made explicit by *assertions*. These assertions play a central part in the Eiffel method for building reliable object-oriented software. They serve to make explicit the assumptions on which programmers rely when they write software elements that they believe are correct. In particular, writing assertions amounts to spelling out the terms of the *contract* which governs the relationship between a routine and its callers. The precondition binds the callers; the postcondition binds the routine.

The Eiffel class in the left column of Figure 1 illustrates assertions (ignore for now the right column). An account has a balance (an integer) and an owner (a person). The only routines – **is** ... **do** ... **end** sequences – accessible from the outside are increasing the balance (deposit) and decreasing the balance (withdraw). Assertions play the following roles in this example.

Routine preconditions express the requirements that clients must satisfy when they call a routine. For example the designer of *ACCOUNT* may wish to permit a withdrawal operation only if it keeps the account's balance at or above the minimum. Preconditions are introduced by the keyword **require**.

Routine postconditions, introduced by the keyword **ensure**, express conditions that the routine (the supplier) guarantees on return, if the precondition was satisfied on entry.

A class invariant must be satisfied by every instance of the class whenever the instance is externally accessible: after creation, and after any call to an exported routine of the class. The invariant appears in a clause introduce by the keyword **invariant**, and represents a general consistency constraint imposed on all routines of the class.

Syntactically, assertions are boolean expressions. To formalize the assertions in our design language, we use a deontic logic based on directed obligations, as

```
class ACCOUNT
feature
 balance: INTEGER
 owner: PERSON
 min_balance: INTEGER is 1000
 deposit(sum:INTEGER) is
  require
   sum >= 0                          Ocr(sum >= 0)
  do
   add(sum)
  ensure
   balance = old balance + sum       Orc(balance = old balance + sum)
  end
 withdraw(sum:integer) is
  require
   sum >= 0                          Ocr(sum >= 0)
   sum <= balance - min_balance      Ocr(sum <= balance - min_balance)
  do
   add(-sum)
  ensure
   balance = old balance - sum       Orc(balance = old balance - sum)
  end
feature [NONE]
 add(sum:INTEGER) is
  do
   balance:=balance+sum
  end
invariant
 balance >= min_balance             Or(balance >= min_balance)
end -- class ACCOUNT
```

Figure 1: class $ACCOUNT$

used in electronic commerce and in artificial intelligence and law [5, 7, 15, 17]. A modal formula $O_{ab}(\phi)$ for a, b in the set of objects (or components, or agents) is read as "object a is obliged toward object b to see to it that ϕ holds". We write c and r for the caller and for the routine, such that the assertions in the program can be expressed as the logical formulae given in the right column in Figure 1. Summarizing:

Require ϕ	$=$	$O_{cr}(\phi)$:	caller is obliged toward routine to see to ϕ.
Ensure ϕ	$=$	$O_{rc}(\phi)$:	routine is obliged toward caller to see to ϕ.
Invariant ϕ	$=$	$O_{r}(\phi)$:	routine is obliged to see to ϕ.

To use these obligations to a deontic design language, we have to add temporal

information. First, we have to formalize **old** *expression*, which is only valid in a routine postcondition. It denotes the value the expression has on routine entry. Consequently, we have to distinguish between expressions true at entry of the routine and at exit of it. More generally, we have to reason how the assertions change over time. For example, the require obligation only holds on entrance, the ensure obligation holds on exit, and the invariant obligation holds as long as the object exists. The obligations only hold conditionally. For example, if the preconditions do not hold, than the routine is not obliged to see to it that the ensure expression holds. Finally, the conditional obligations come into force once the object is created, and cease to exist when the object is destructed.

We therefore combine the logic of directed obligations with linear time logic (LTL), well known in specification and verification [8]. There are many alternative temporal logics which we could use as well. For example, in [4] deontic logic is extended with computational tree logic in $\mathrm{BDIO}_{\mathrm{CTL}}$. Semantics and proof theory are straightforward, see for example [4]. Due to space limitations, we do not give the details. Instead, we address the question how to use the logic to reason about assertions.

Definition 1 (Syntax $\mathbf{O_{LTL}}$) *Given a finite set A of objects (or components, or agents) and a countable set P of primitive proposition names. The admissible formulae of O_{LTL} are recursively defined by:*

1 Each primitive proposition in P is a formula.

2 If α and β are formulae, then so are $\alpha \wedge \beta$ and $\neg\alpha$.

3 If α is a formula and $a, b \in A$, then $O_{a,b}(\alpha)$ is a formula as well.

4 If α and β are formulae, then $X\alpha$ and $\alpha U \beta$ are formulae as well.

We assume the following abbreviations:

$$\begin{array}{rcl@{\qquad}rcl} \alpha \vee \beta & \equiv_{def} & \neg(\neg\alpha \wedge \neg\beta) & \alpha \rightarrow \beta & \equiv_{def} & \neg\alpha \vee \beta \\ \Diamond(\alpha) & \equiv_{def} & \top U \alpha & \Box(\alpha) & \equiv_{def} & \neg\Diamond(\neg\alpha) \\ O_a(\alpha) & \equiv_{def} & O_{a,a}(\alpha) & & & \end{array}$$

We assume the following propositions: create(c) holds when object c is created, destruct(c) holds when object c is destructed, call(c_1,c_2,f) holds when object c_1 calls routine f in object c_2. We assume that if a routine in an object is called, there is an earlier moment in time at which the object is created. However, since our operators only consider the future, this property cannot be formalized. We assume that propositions can deal with integers, a well known issue in specification and verification, see [8] for further details. Finally, we assume that the time steps of the temporal model are calls to routines. The first routine and the invariant in the example can now be formalized as:

$$\begin{array}{l} \text{call}(c_1,c_2,\text{deposit(sum:INTEGER)}) \rightarrow O_{c_1,c_2}(\text{sum} >= 0) \\ (\text{call}(c_1,c_2,\text{deposit(sum:INTEGER)}) \wedge (\text{sum} >= 0) \wedge (\text{balance} = \text{b})) \rightarrow \\ \qquad\qquad XO_{c_2,c_1}(\text{balance} = \text{b} + \text{sum}) \\ \text{create}(c) \rightarrow (O_c(\text{balance} >= \text{min\_balance}) \text{ U } \text{destruct(c)}) \end{array}$$

3 Contract violations

Whenever there is a contract, the risk exists that someone will break it. This is where exceptions come in. Exceptions – contract violations – may arise from several causes. One is an assertion violation, if run-time assertion monitoring is selected. Another is a signal triggered by the hardware or operating system to indicate an abnormal condition such as arithmetic overflow, or an attempt to create a new object when there is no enough memory available. Unless a routine has been specified to handle exceptions, it will **fail** if an exception arises during its execution. This in turn provides one more source of exceptions: a routine that fails triggers an exception in its caller.

A routine may, however, handle an exception through a **rescue** clause. An example using the exception mechanism is the routine *attempt_deposit* in Figure 2 that tries to add sum to balance. The actual addition is performed by an external, low-level routine *add*; once started, however, *add* may abruptly fail, triggering an exception. Routine *attempt_deposit* tries the deposit at most 50 times; before returning to its caller, it sets a boolean attribute *successful* to *True* or *False* depending on the outcome. This example illustrates the simplicity of the mechanism: the **rescue** clause never attempts to achieve the routine's original intent; this is the sole responsibility of the body (the **do** clause). The only role of the **rescue** clause is to clean up the objects involved, and then either to fail or to retry.

```
attempt_deposit(sum:INTEGER) is
 local
  failures: INTEGER
 require
  sum >= 0;                          Ocr(sum >= 0)
 do
  if failures < 50 then
   add(sum); successful := True
  else
   successful := False
 rescue
  failures := failures + 1; retry
 ensure
  balance = old balance + sum        Orc(balance = old balance + sum)
 end
```

Figure 2: Routine *attempt_deposit*

The principle is that *a routine must either succeed or fail*: it either fulfills its contract, or not; in the latter case it must notify its caller by triggering an exception. The optional **rescue** clause attempts to "patch things up" by bringing the current object to a stable state (one satisfying the class invariant). Then it can terminate in either of two ways:

- The **rescue** clause may execute a **retry** instruction, which causes the rou-

tine to restart its execution from the beginning, attempting again to fulfil its contract, usually through another strategy. This assumes that the instructions of the **rescue** clause, before the retry, have attempted to correct the cause of the exception.

- If the **rescue** clause does not end with a **retry**, then the routine fails; it returns to its caller, immediately triggering an exception. (The caller's **rescue** clause will be executed according to the same rules.)

In our design language, the exception can be formalized as a violation, and the exception handler gives rise to a so-called contrary-to-duty obligation. For example, there is a violation if the postcondition does not hold, i.e., we do not have balance = old balance + sum. In case of violation, a retry means that the obligation persists until the next time moment. We extend the language with the proposition *retry*. Now, the fact that a retry implies that the postcondition holds again for the next moment can be characterized as follows:

$$O_{c_1,c_2}(\phi) \wedge \neg\phi \wedge \mathit{retry} \rightarrow XO_{c_1,c_2}(\phi)$$

4 The normative stance

In this section we compare the normative stance, a phrase due to Jan Broersen [2], implicit in the design by contract with the intentional or BDI stance popular in agent based software engineering. Table 1 summarizes the comparison between the intentional stance and the normative stance.

Stance	intentional stance	normative stance
Concepts	BDI	OP, rights, responsibility
from	folk psychology	ethics, law, sociology
Computer	human = angry, selfish, ...	God, master/slave, servant
Class of systems	decision making	decision making
Realization	specification and verification	components
Implementation	programming	objects, operation
specification	BDI_{CTL}	temporal deontic logic

Table 1: intentional vs normative stance

First, the intentional stance is rooted in the philosophical work of Dennett, whereas such grounding does not seem to exist for the normative stance (though there are candidates, such as [1]). The concepts from the intentional stance come from folk psychology. The normative stance borrows concepts from ethics, law or sociology. Other examples of this normative stance we mentioned in the introduction are the IMPACT system [6] and the BOID architecture [3].

Second, the success of the intentional stance is that people like to talk about their computer as a human which has beliefs and desires, which may be selfish, or which can become angry. The implicit assumption of design by contract is that designers find it useful to understand software construction in terms of contracts,

or, more generally, in terms of obligations. The success is due to the fact that humans either consider the computer as their master, which has to be obeyed, or as their slave, which has to obey orders.

Third, the intentional stance has been advocated for agent systems, which are for example autonomous and proactive. It has been used as a high level specification language, as well as low level programming language. We believe the normative stance can be used in a wider setting. In the examples we used it for low level objects. However, it is particularly useful if we use a higher abstraction level in terms of components or agents.

5 Concluding remarks

In this paper we study how deontic logic can be used as a design language to support design by contract. First, we ask which kind of deontic logic can be used as a design language to support design by contract We show how directed modal operators are capable of formalizing contracts between clients (callers) and and suppliers (routines), relying on mutual obligations and benefits made explicit by assertions. These formalisms have been developed and studied in electronic commerce and artificial intelligence and law. Moreover, we show how temporal operators can be used to formalize dynamic behavior such as contract violations.

Second, we ask what kind of properties should be formalized by such a design logic. This is summarized in Table 2. In this paper, we do not consider contract forms and contracts for testing and debugging. The contract form of a class, also called its "*short form*", serves as its interface documentation. It is obtained from the full text by removing all non-exported features and all implementation information such as **do** clauses of routines, but keeping interface information and in particular assertions. The use of these elements in our deontic design language, for example to *combine* assertions, is subject of further research.

social contract	assertions	directed obligations
violation	exception	violations
repair	exception handling	contrary-to-duty reasoning
contract form	interface	?
testing and debugging	?	?

Table 2: Bridging the gap

Third, we ask how this approach based on deontic logic compares to the BDI approach, dominant in agent based software engineering. Whereas the BDI approach is based on an attribution of mental attitudes to computer systems, design by contract is based on an attribution of deontic attitudes to systems. We suggest that the normative stance has a wider scope of applicability than the intentional stance, though this has to be verified in practice. In further research we study the relation with commitments in Shoham's Agent Oriented Programming (AOP) [14], and with rely/guarantee reasoning [16].

References

[1] R. Brandom. *Making it explicit*. Harvard University Press, Cambridge, MA, 1994.

[2] J. Broersen. *Modal Action Logics for Reasoning about Reactive Systems*. PhD thesis, Vrije Universiteit Amsterdam, 2003.

[3] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.

[4] J. Broersen, M. Dastani, and L. van der Torre. BDIO_{CTL}: Properties of obligation in agent specification languages. In *Proceedings of IJCAI'03*, pages 1389–1390, 2003.

[5] F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7(1):69–79, 1999.

[6] T. Eiter, V. Subrahmanian, and G. Pick. Heterogeneous active agents, I: Semantics. *Artificial Intelligence*, 108:179–255, 1999.

[7] C. Krogh and H. Herrestad. Hohfeld in cyberspace and other applications of normative reasoning in agent technology. *Artificial Intelligence and Law*, 7(1):81–96, 1999.

[8] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, Heidelberg, Germany, 1992.

[9] B. Meyer. Invitation to Eiffel. Technical Report TR-EI-67/IV, Interactive Software Engineering, 1987.

[10] B. Meyer. Design by contract. In D. Mandrioli and B. Meyer, editors, *Advances in Object-Oriented Software Engineering*, pages 1–50. Prentice-Hall, New York, London, 1991.

[11] B. Meyer. Applying design by contract. *IEEE COMPUTER*, 25(10):40–51, 1992.

[12] B. Meyer. Systematic concurrent object-oriented programming. *Communication of the ACM*, 36(9):56–80, 1993.

[13] J. Meyer and R. Wieringa. *Deontic Logic in Computer Science: Normative System Specification*. John Wiley and Sons, 1993.

[14] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[15] M. P. Singh. An ontology for commitments in multiagent systems: toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113, 1999.

[16] E. W. Stark. A proof technique for rely/guarantee properties. In *Foundations of Software Technology and Theoretical Computer Science*, volume 206 of *Lecture Notes in Computer Science*, pages 369–391, 1985.

[17] Y. Tan and W. Thoen. Modeling directed obligations and permissions in trade contracts. In *Proceedings of the Thirty-First Annual Hawaian International Conference on System Sciences*, 1998.

[18] G.H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

[19] R. Wieringa and J. Meyer. Applications of deontic logic in computer science: A concise overview. In *Deontic Logic in Computer Science*, pages 17–40. John Wiley & Sons, Chichester, England, 1993.

User-profiles for Information Retrieval

Bas van Gils
basvg@cs.kun.nl

Eric D. Schabell
ericS@cs.kun.nl

University of Nijmegen, Computing Science Institute, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands

Abstract

In this position paper we will investigate a novel architecture for profile-based retrieval on the Web called *Vimes*[1]. This architecture is based on the fact that resources found on the Web should not only be topically relevant to a searcher's query; other characteristics (such as the file format or structural format) of a resource are equally important. Furthermore, *Vimes* uses profiles to deal with user characteristics and constraints.

1 Introduction

In today's "information society" information plays an increasingly important role. The trick is to get the right information at the right time and in an appropriate format for a given goal. Finding the right information has been researched extensively in the IR-field over the last decades. Already in the 1970's people tried to devise computer programs to assist them in their search for information. These computerized searches started with searching in homogeneous document collections such as in the STAIRS-project (Salton and McGill, 1983). The search process became more elaborate with the apparent rise of the Web. It led to the introduction of search engines such as GOOGLE which not only indexes (hyper)text, but also images, PDF-documents and interactive databases such as Citeseer (Citeseer, 1997). In other words, search engines attempt to retrieve relevant *resources*, rather than documents alone.

The importance of the timing aspect is particularly obvious when investment decisions are involved, such as on the stock market. Getting some information late could have huge (financial) consequences. Implementing a strategy for getting information in time often depends on many things such as choosing the right partner/supplier: some news sites are 'faster' then others in picking up news.

The third aspect mentioned deals with formats in the broad sense. It refers to "file format" (e.g. PDF, or HTML) as well as "structural format" (e.g. "abstract",

[1] Vimes is the Commander of the City Watch in Terry Pratchett's Discworld series™. We hope that our system will show the same ingenuity, tenacity and have the same success in information retrieval as Vimes has had at solving crimes.

or "photograph")[2]. The file format issue has been around since the early days of computing. Since people use different tools for jobs such as text processing a need for conversion tools between the file formats arose. Many of these conversions are available today. This is not (yet) the case for the latter issue, even though attempts have been made. A good example of this type of software is a computer program that generates abstracts for expository text (see e.g. (Barzilay and Elhadad, 1997)).

It is apparent that these factors vary for different users of IR-systems. For some users it is ok if certain financial records arrive slightly late, whereas for others it might have unpleasant consequences, some people would prefer an abstract of a (large) report over its full text etcetera. In other words, each of these factors can be seen as a *characteristic* of a searcher. Loosely defined, a *profile* is the collection of all characteristics of a searcher that are relevant for the retrieval process. The goal of this position paper is to investigate how profiles can be used to improve the IR-process and define the architecture of `Vimes`.

2 Overview

One of the basic functions of any information retrieval (IR) system is *relevance ranking*: the (characterizations of) resources are ranked such that the resources that are "most relevant" are listed first, and the ones that are least relevant are listed last. In (Dhyani et al., 2002) an overview is given of metrics that are used to determine the relevancy of a Web-document with regard to a query. Furthermore, it is pointed out that relevancy involves more than *topical relevance*; other attributes of resources (such as its quality and price) are important as well.

2.1 Relevance

In (Gils et al., 2003) a *conceptual model* for information supply is presented. This model is based on the notion that similar information can be conveyed by multiple representations, leading to the notion that several representations (resources on the Web) can belong to a single information service (provide access to their underlying representations). Based on this work, we define:

Definition 2.1 (representation format) *It is enforced that each representation has exactly one type. Examples of these types are:* PDF, HTML *and Webservice.*

Definition 2.2 (structural format) *Not all representations that belong to a single information service have to convey the same amount of information. For example, one conveys the "full content" and another is merely an "abstract". These (types of) structural format are modeled as* feature types *in (Gils et al., 2003). In this article we refer to them as the structural format.*

Using these definitions we can introduce our notion of relevance. Apart from topical relevance, which is the 'traditional' way of measuring relevance, we define that other constraints must be met as well. Examples of such constraints are

[2] In (Gils et al., 2003) the *structural format* is modeled as *feature types*.

its format (as explained in the previous section), but also price, quality etcetera. It may very well be that a searcher is willing to pay a certain amount of money in order to get his hands on a high-quality resource! Hence, we define relevance as follows:

Definition 2.3 (Relevance) *Resources are relevant with regard to a query if and only if this resource meets all the criteria that a searcher poses on it. These criteria can be formulated in either the query, or the user-profile.*

This definition resembles the notion of functional versus non-functional requirements in Software Engineering (Sommerville, 1989). It is now well accepted that non functional requirements and functional requirements are equally important to any software engineering project (see e.g. (Cysneiros and do Prado Leite, 2002; Barrett, 2002) for a discussion on the importance of non functional requirements).

This modified view of relevance has an impact on *precision* and *recall*, for it is 'less easy' for a document to be relevant with regard to a query. For example, it may be that a resource must be converted to another format before it is really relevant. In Section 3 we explain how a retrieval system can exploit this new notion of relevance in order to achieve 'better retrieval'.

2.2 Profiles

Already in (Myaeng and Korfhage, 1986) it was recognized that information retrieval systems can be personalized for users by means of profiles. During the last few decades a lot of research has been invested in the area of user profiles. Often, these profiles are used to enhance the query by capturing the user's notions of query terms (see e.g. (Myaeng and Korfhage, 1986; Chen and Kuo, 2000; Pierra et al., 2000)). However, profiles can be used more extensively. For example, in (Gligor, 1996) profiles are used for access control. We define that:

Definition 2.4 (Profile) *A (user) profile consists of a set of preferences with regard to behavior of a search engine as well constraints on the results it presents to the user.*

To illustrate this definition, the following list are the items that make up a particular user-profile:

preferences : I prefer a maximum of 25 results per page, and by selecting a relevant resource (clicking on the link) will open a new window.

constraints : I prefer HTML and PDF formats and refuse the Microsoft DOC-format. Furthermore, the size of the resource should not exceed 25Mb.

Using this definition, there are two areas in the retrieval process where profiles can be used. Firstly, they can be used for *post-processing* the results of the ranking process. For example, an resource that was found to be topically relevant can be converted to the proper format (See Section 2.1). Furthermore, profiles can be used to make sure that the retrieval engine operates according to the user's wishes.

2.3 Format

In the previous section we explained what profiles are and what they can be used for. In this section we present a *possible* format for storing these profiles, whereas in the next section we explain how/where they are stored exactly.

Since we want the profiles to be re-used across (Web) search engines, the format should be an *open standard*. More specifically, we want our format to be machine understandable and interoperable. The eXtensible Markup Language (XML, see e.g. (Bray et al., 2000)) is particularly well suited for this task (see e.g. (Suryanarayana and Hjelm, 2002)). The following XML-fragment is an example of what a profile could look like:

```
<? xml version="1.0" ?>
<!-- ------------------------------------------------------------ -->
<!-- A profile has an owner, identified by his/her Email-address. -->
<!-- Furthermore, a check-sum is included for security purposes.  -->
<!-- This profile stores 3 characteristics.                       -->
<!-- ------------------------------------------------------------ -->
<!-- define the owner of the profile -->
<profile owner="Bas van Gils" email="bas.vangils@cs.kun.nl" cs="2768A493">
    <!-- 1st characteristic: how many results per page? -->
    <characteristic type="results"> <page> 25 </page> </characteristic>

    <!-- 2nd charactersitic: the max. size in Mb -->
    <characteristic type="max_size"> <mb> 5 </mb> </characteristic>

    <!-- 3rd characteristic: preferred file-types -->
    <characteristic type="file_type">
        <type nr="1"> HTML </type>
        <type nr="2"> PDF </type>
        <type nr="3"> PS </type>
    </characteristic>
</profile>
```

Note that this excerpt is intended to illustrate our ideas. Defining a formal DTD for profiles is part of future research.

3 Architecture

In the previous sections we explained our notion of formats, profiles and relevance. These notions are essential for the architecture of *Vimes*, which we will introduce here. The architecture uses many elements that stem from previous research, such as brokers, agents, semantic web components and web services.

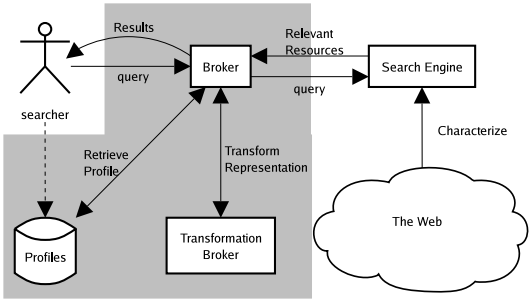
3.1 Components

User profiles will be stored in a repository for easy access and reuse by *Vimes*. The format in which this information will be maintained should be an accepted open standard, such as XML as outlined briefly in the previous section. Using such an open standard will make life easier for the user as they only need to define their preferences once. All retrieval engines that know how to deal with (this type of) profiles can re-use this single profile[3].

[3]Open issues that we need to work out still are the management of these profiles: where to store them? how to achieve an acceptable level of security?

In the previous section we introduced a new notion of relevance. We also explained how, in some cases, resources may have to be transformed before they are considered to be relevant. To cater for these transformations we introduce a *transformation broker*[4]. The broker will be a networked service, encapsulating functionality of all available transformation tools on the network and provide for multiple methods of transport. For example, a request to the transformation broker includes the form desired is PDF and the resource document is a postscript document. This particular conversion can be achieved by a transformation broker on the network that provides the tool *ps2pdf*. For similar reasons as before, we choose open standards for transport, such as FTP and HTTP. An additional benefit is that other parties can more easily participate/contribute by submitting transformation routines to the broker.

The broker component will be `Vimes`' main interface for users seeking information. It will interact with the user-profile repositories and search engines on the Web. Essential to our architecture is the broker's ability to interact with not only the well known web search engines (Yahoo, GOOGLE, AltaVista, Excite, etc.), but also with such enabling technologies as static agents, mobile agents, web services and services using the Semantic Web or Resource Description Framework (see e.g. (Google, 2003; Fünfrocken and Mattern, 1999; Berners-Lee et al., 2001; Miller et al., 2003)). Our broker component will also provide interaction with different forms of user-profile repositories, both local and remote. This will allow interaction with other profile systems on the Web (see e.g. (Pierra et al., 2000) for an agent-based approach along these lines). This leads to the following architectural diagram, with the components in the shaded area making up the `Vimes`-system.



Please note that the components will be loosely coupled so that they (especially the profile repository and the transformation broker) can also be accessed by other systems via the Web.

3.2 Example session

In this section we describe what the retrieval process could look like, based on the architecture as defined in the previous section. The first thing to be done is that the user creates a profile, preferably via an intuitive Web-interface, after which

[4]This transformation broker flowed out of the earlier work done on resource access for generic information retrieval (Schabell, 2002).

it can securely be stored in the repository. The second step is to browse to the broker, which functions as the main interface for the rest of the process. The user identifies himself (either automatically via e.g. a cookie, or more explicitly via a login-screen) after which the relevant profile is retrieved from the repository.

When the profile is retrieved, the user can enter his query into the system. Two things can happen at this point: either the broker decides to reformulate the query based on the user-profile, or it leaves the query untouched. Subsequently, the query is submitted to one of the search engines. This can be one of the well known web search engines, but others are possible such as an agent, a web service or other external services as described above. Based on the user's profile, the broker may decide to post-process discovered resources. The returned list of discovered resources would then be transformed using the transformation broker. If this is indeed the case, the resources are processed and ranked again before they are presented to the user.

3.3 Future work

The goal is to implement this architecture within the PRONIR research project (Proper, 2002) in the coming years. We will have to look deeper into the enabling technologies and will have to make a decision to narrow the possibilities down for our prototype. In either case, we will attempt to use open standards (XML, SOAP, WSDL, UDDI, etc) and open protocols (HTTP, FTP, Jabber, etc) throughout the entire architecture. Currently we are under way with regards to the transformation broker. We have setup a Conversion Clearinghouse that is web accessible, allowing users to search through our available conversions. In the near future we hope to have this online for extern input and usage. Development continues on this part of the project (Schabell, 2003).

4 Conclusion

In this article we started out by introducing a new way of measuring relevance. We feel that the relevancy of a resource with regard to a user query should be a combination of several factors such as topical relevance (the traditional measurement), but also things like its format, price, quality etcetera. To be able to deal with this more elaborate relevancy metric, we need to know a lot more about the user. Part of this additional knowledge is fixed over time for individual users. For example, a search may *always* prefer a certain file-type over others, and wants the search engine to list a maximum of 25 results per page. This lead to the introduction of profiles, which consist of constraints on the resources that are presented to the user as well as preferences with regards to the behavior of the search engine.

The main contribution of this article is the architecture of `Vimes`, introduced in Section 3. `Vimes` is intended to be a broker that assists users in querying the Web. There are three important components in this architecture. The *profile repository* stores the profiles of all users in an open format such as XML. There are

still some open issues in this area, such as specifying a language for storing the profiles, enforcing that they are stored securely, etcetera.

The second component is the *transformation broker*, which enables us to perform transformations on resources found on the Web. With these transformations we hope to be able to transform resources into a format that is convenient / wanted by individual users. For example, we can transform a HTML document into PDF, or generate an abstract of a report that is too long according to a user's profile. We are currently working on a system that performs these transformations by setting up a Conversion Clearinghouse that is web accessible, allowing users to search through our available conversions.

Last but not least, the *broker* in the `Vimes` architecture is the user-interface. It interacts with the two other components, as well as with search engines on the Web. Much work remains to be done in this are also. For example, we need to figure out what the interface will look like, which message-standards are going to be used to interface with the other components, etcetera.

All in all, this article provides insight into a novel way of thinking about retrieval. It outlines the architecture `Vimes`, without giving a full specification. Finally, we have presented a road-map for our research.

References

Barrett, M. L. (2002). Putting non-functional requirements to good use. *The Journal of Computing in Small Colleges*, 18(2):271–277.

Barzilay, R. and Elhadad, M. (1997). Using lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97)*, Madrid, Spain.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web, a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., and Maler, E. (2000). Extensible markup language (XML) 1.0 (second edition). Technical report, World Wide Web Consortium,
`http://www.w3.org/TR/REC-xml`. last checked: 19-may-2003.

Chen, P.-M. and Kuo, F.-C. (2000). An information retrieval system based on a user-profile. *The Journal of Systems and Software*, 54(1):3–8.

Citeseer (1997). *NEC ResearchIndex Citeseer*.
`http://citeseer.nj.nec.com`. Last checked: 19-may-2003.

Cysneiros, L. M. and do Prado Leite, J. C. S. (2002). Non-functional requirements: from elicitation to modelling languages. In *Proceedings of the 24th international conference on Software engineering*, pages 699–700, Orlando, Florida. ACM Press. ISBN: 1-58113-472-X.

Dhyani, D., Ng, W. K., and Bhowmick, S. S. (2002). A survey of web metrics. *ACM Computing Surveys (CSUR)*, 34(4):469–503. ISSN:0460-0300.

Fünfrocken, S. and Mattern, F. (1999). Mobile agents as an architectural concept for internet-based distributed applications - the wasp project approach. In Steinmetz, editor, *Proceedings of the KiVS'99 ("Kommunikation in Verteilten Systemen")*, pages 32–43. Springer-Verlag.

Gils, B. v., Proper, E., and Bommel, P. v. (2003). Towards a general theory for information supply. In *Proceedings of the 10th International Conference on Human-Computer Interaction*.

Gligor, V. (1996). Characteristics of role-based access control. In *Proceedings of the first ACM Workshop on Role-based access control*, Gaithersburg, Maryland, United States. ACM Press. ISBN: 0-89791-759-6.

Google (2003). *Google Web API's*. Google, `http://www.google.com/apis`. last checked: 16-May-2003.

Miller, E., Swick, R., and Brickley, D. (2003). *Resource Description Framework (RDF)*. World Wide Web Consortium, `http://www.w3.org/rdf`. last checked: 16-May-2003.

Myaeng, S. H. and Korfhage, R. R. (1986). Towards an intelligent and personalized retrieval system. In *Proceedings of the ACM SIGART international symposium on Methodologies for intelligent systems*, pages 121–129, Knoxville, Tennessee, United States. ACM Press. ISBN:0-89791-206-3.

Pierra, S., Kacan, C., and Probst, W. (2000). An agent-based approach for integrating user profiles into a knowledge management process. *Knowledge-Based Systems*, 13(5):307 – 314.

Proper, E. (2002). PRONIR proposal. Technical report, IRIS / KUN. NWO Project Proposal.

Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill New York, NY.

Schabell, E. D. (2002). Resource access in generic information retrieval systems. Master's thesis, Vrije Universiteit, Amsterdam, Netherlands.

Schabell, E. D. (2003). *Profile Based Retrieval Of Networked Information Resources, The Scientific Programmers Workshop*. `http://www.pronir.nl/pub/spws`. Last checked: 16-May-2003.

Sommerville, I. (1989). *Software Engineering*. Addison-Wesley, Reading, Massachusetts.

Suryanarayana, L. and Hjelm, J. (2002). Profiles for the situated web. In *Proceedings of the eleventh international conference on the World Wide Web*, pages 200–209, New York, NY, USA. ACM Press. ISBN:1-58113-449-5.

Concepts and navigation targets

Stephan ten Hagen

University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
stephanh@science.uva.nl

Abstract

In this paper we describe a framework in which concepts are related to targets of navigation tasks. Concepts are represented by policies, that are improved based on evaluations of a human teacher. Eventually, when the task can be completed, actions with high expected rewards indicate which observation show the 'concept'. In an experiment we show that the robot is capable of selecting observations that show the target concepts.

1 Introduction

In every day life we get surrounded more and more by 'smart' utilities, for which we have to specify what they have to do for us. For video recorders, the program itself is fixed and the user has to set parameters, like time and channel, used by the program. The more complicated machines become, the harder it will be for the human user to make machines do exactly what they want them to do. Already many people have difficulties with programming a video.

In this paper we look into a way of making it easier for human users to 'program' their equipment. Rather that teaching the user how to program, we will make the equipment learn what should be done. For this, the user has to specify the task which the user probably expresses using concepts that have meaning for himself, but not necessary for the equipment. For an unknown concept, all the equipment can do is try something. The user can evaluate these trials and based on these evaluations, the equipment can learn the meaning of the concept and the corresponding task that should be executed.

In this paper we use a mobile robot with a navigation tasks as test application. Concepts will be in the form of "Go to the door", "Go to the corner", or "Go to John's office". The "go to" is already implied, because it is the only thing the mobile robot can do. The targets "door", "corner" and "John's room" are the concepts that the robot has to learn to recognize using its sensor. These concepts are names used by the user and do not have to correspond to particular sensory observations. Somehow the robot has to label observations, such that they correspond with the area the users associates with the concept.

In the next section we provide some background information. In section 3 we present our framework to select observations that correspond to the targets of the task. In section 4 we demonstrate this in an experiment.

2 Background

The word 'concepts' itself is rather vague and is used in many different situations. We will consider concept learning as a task whose objective is too learn to discriminate between items according to some underlying rule. This can be a database with cases of which a subset is manually labeled to belong to a certain category. The learning task is then to determine which specific attributes of these cases can be use to distinguish them from the rest. This is not possible without the labels. If these labels are not available concept formation can be used to dynamically categorize the cases through interactions.

2.1 Concepts formation

A well know example of concept formation is the "Talking Heads Experiment", a project where a 'language game' is performed on many different robots [9]. The general idea of language games, or 'guessing games', is that agents transmit 'words' about the environment. One agent is the speaker and the other the listener and the objective of the game is that the listener guesses the correct meaning of the word transmitted by the speaker. The correct meaning here refers to the ability of the listener to identify the exact same object in the environment the speaker associates with the transmitted word. If the listener guesses correctly then both the speaker and listener reinforce that category.

One motive of research in language games is that they may explain the origin of language or intelligence [8] in the sense that a shared lexicon emerged among a group of interacting individuals. Such lexicons can be found through associations between the perceived situations together with the received signals (words) [3][4]. This is a categorization of the input space. By dividing the input space in a finite discrete set, the signals can be 'grounded' to the partitions of the sensory input space. This resembles determining relevant attributes of cases in concept learning, when the perceptions are viewed as cases.

Using only categories of perceptions has a rather limited use. Agents are only capable of recognizing concepts when placed in their view. They cannot actively contribute to the concepts formation or interact with the environment. In [14] concepts are related to the sensorimotor control. Dynamics is included by adding a finite state automaton. Now the listener has to guess the current state of the speaker and predict its action. In this way concepts formation is more related to control and manipulating the environment, then just perceiving and trying to understand the environment.

2.2 Behavior based robotics

The ideas of guessing games are also investigated for other robot tasks, like target tracking [13]. No longer does the origin of language form the main objective of research. It starts to resemble a behavior based robotics approach. The underlying idea of behavior based robotics is that the control architecture consist of a set of low level reactive controllers. These low level controllers compete with each other

and a selection mechanism exist to choose the controller that determines the action applied to the actuators. A good selection mechanism makes that reasoning is no longer needed [2], but if it is not available it can be learned. In [5] an approach is shown where Reinforcement Learning (RL) is applied to optimize the selection. The representation used is referred to as a behavior based network. In [7] this network is used in the context of training by example, where the trainer can be a human or an other robot.

A possible definition of a language is given in [6] as: "a suggestion by objects, actions or conditions of associated ideas or feelings". The work is similar to that in [7] but now also implicit communications is included. The implicit communication is the communication through actions (see the language definition). A human observer is capable of understanding the intentions from the exhibited behavior, because the action carry the intentional meaning and are therefore easy to understand. The sequence of actions is the robot's way of saying "I think you meant this". If this was meant by the human observer, the robot can be rewarded to reinforce that concept.

The behavior based approaches differ from the guessing game approaches in the sense that they do not assume two agents that have to start from scratch and find an agreement. It is more a teacher/pupil configuration where one agent is already familiar with the concepts and the other has to learn it. Because the teacher cannot know how the other agent perceives the environment, it has to observe its behavior and judge that instead.

3 Framework

In our framework concepts are related to targets of navigation tasks. For each task actions are executed according to a specific policy that is greedy with respect to a corresponding value function. These value functions are formed by associating values, representing the expected sum of future reinforcement, to observations taken in the environment. So each policy is anchored in the environment through the observations and represents the concept (goal of the navigation task). We will use a mobile robot with omnidirectional camera to introduce this framework.

3.1 Low level reactive behaviors

Low level reactive behaviors can be implemented as regulators that use the error between the current pose and some target pose as input to compute control actions. This requires that the robot keeps track of its location and orientation. Visual homing is a visual navigation method where the target 'pose' is indicated by a target image. In [1] it is applied to a camera mounted on a robot arm. If selected 'interesting' features in the current and target image are combined to form corresponding pairs, the difference in position of these pairs can be used to estimate the relative difference in pose at which the images were taken.

Images taken by a mobile robot with an omnidirectional camera can be used as target images. In an area around the pose where an image was taken, the robot is capable of estimating the relative difference in pose. This is the error needed

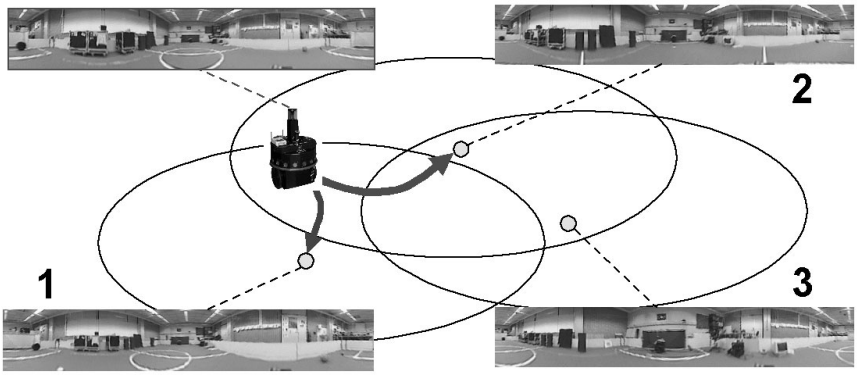


Figure 1: Navigation example: The robot can reach 1 and 2 but not 3. When 2 has a higher value the robot will move to the place where image 2 was taken. At a certain point 3 may be reachable and if 3 has a higher value the robot changes target image.

by the regulator and therefore the robot can find the target pose without actually knowing its current pose. There is no need to keep track location and orientation in the area where an image was taken. Images taken further away from the target pose will be too dissimilar such that the robot is no longer able to home in to the target.

3.2 Navigation

A robot that moves around and stores each observations, generates a database with potential target images. As described in section 2, a behavior based approach assumes a set of competing low level controllers. In our framework there is only one low level controller with a finite set of potential targets images in the database. The images are competing as targets. However, not all images will be competing, only those taken near the robots current pose.

The areas in which images in the database can be reached should overlap a little, such that the robot will have a choice in target image at any pose. The policy is to choose the target image that has the highest value associated with it. In figure 1 an example of the navigation behavior is shown. The areas in which target images can be reached should not be to large. Otherwise one image with a large area and a high value can make many images in the database obsolete. In [11] it is shown that for our setup the radius of areas is about 2 meters.

Note that this navigation method is not restricted to omnidirectional images. Any sensor for which the relative difference in pose can estimated from two observation will do.

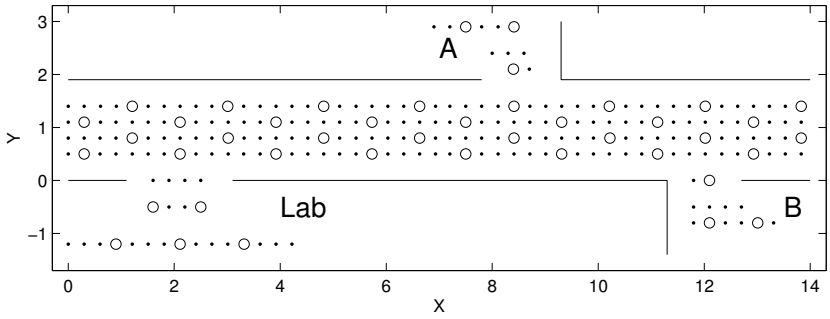


Figure 2: The data: This shows our corridor and parts of the lab, room A and room B. All dots and circles are poses where an image was taken. The circles represent the images in the database.

3.3 Learning

Instead of manually assigning values to images, these values can also be learned. As indicated before in [12], the overlapping areas around images can be regarded as a coarse code representation [10] of the continuous state space. This representation is often used in reinforcement learning, and provides the ability to train the robot using rewards and punishments. This is exactly what we need, if we want the user to be able to train the robot by evaluating its behavior.

Note that the values associated with the images form the policy because they determine the selection of the targets. This implies that the policy is anchored to the environment. If the robot can 'go to the door' in one room, it may not be able to do that in an other room. This is because the robot has learned what the the correct behavior is, but it hat not learned why this is the correct behavior. Concept learning can be used to find out why, if the value is used to add labels to the images in the database. High values correspond with a high expected rewards, so images with high values are most likely taken in the area corresponding to the concept. The specific attributes found by concept learning then describe the actual concept in terms of perception.

4 Experiment

The purpose of the experiment is to demonstrate that the framework described in the previous section is capable of selecting a set of observations that correspond to the target of the task.

4.1 Setup

We generated a data set of 234 images, taken mostly in our corridor, but some were taken at the entrance of the Lab and two other rooms (A and B). We measured manually the poses at which images were taken. We selected 43 target images

to form the database with potential targets. The rest was used to simulate the movements of the robot. In this way we could repeat experiments under exactly the same conditions. Note that we used the measured poses *only* for the simulation of the robot movements. Figure 2 shows where images were taken and which images are selected as targets.

4.2 Results

All images were transformed to 720 by 120 panoramic grayscale images and features were selected according to [11]. Then we compared each image with a target image and estimated the relative difference in pose according to [11]. If this was possible, then this target could be selected from this pose. The average number of possible targets for a pose was 4.95. To simulate the robot's movements the low level controller selects the neighbor pose that is closest to the target.

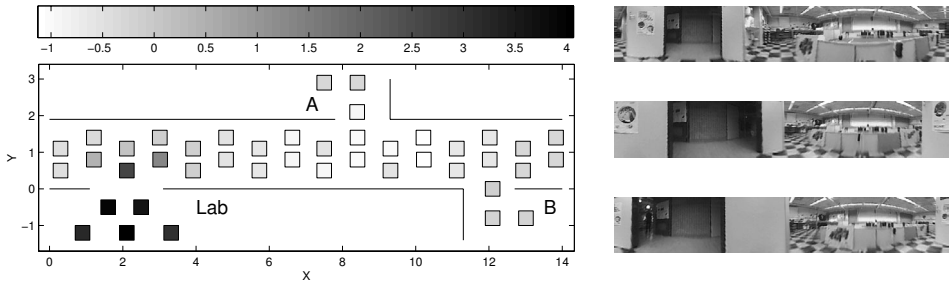
We did experiments with the Lab, room A and room B as target, and so we created 3 policies. Because we simulated the motion of the robot we also had to simulate the human teacher. A punishment was given in form of a -1 reinforcement, when the robot moved away from the door of the target area. A reinforcement of 3 was given as reward for being at the target area. Because punishments are given directly for an incorrect action, we had to use TD(0) learning. We only associate one value with each target and not four like in [12]. These values were initialize between -10^{-7} and 10^{-7}.

We explored by ranking all possible targets by their value and ignore each of them with the same probability. We then picked from the remaining targets the one with the highest value. The result is that the target with the highest value has the highest probability of being picked, the second has the second and so on. This kind of exploration resembles consequences of sensor uncertainty, when assuming that each target has the same probability of being missed.

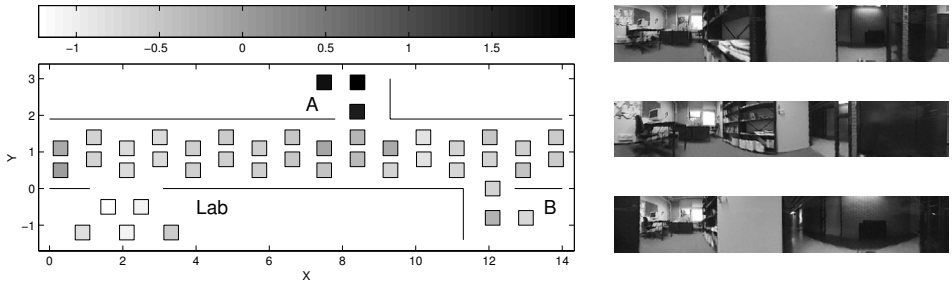
We did 50 runs of 100 steps for each target concepts. After each run the robot started at a random pose in the environment. In figure 3(a) we see the result for 'go to the lab'. The significantly higher values in and near the robot lab is clear. The same hold for the results in figure 3(b) and figure 3(c) Figure 3 also shows the three images with the highest value for each task. These images clearly show the robot's view when entering these rooms. Only the behavior of the robot and evaluation of the teacher were used to select these images. Now they can be used to describe the perceptual concept for each task.

5 Conclusion

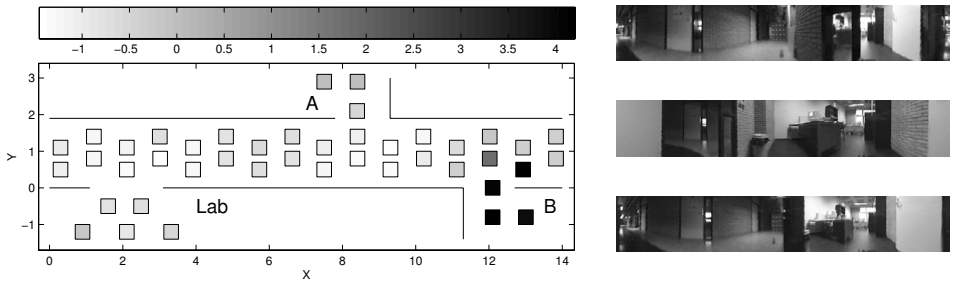
In this paper we introduced a framework where concepts were learned by first learning a policy, that can be used to show that the concept was understood. For a mobile robot with omnidirectional camera we have shown that the navigation task can be learned and that observations corresponding to the concept can be selected using the resulting value function.



(a) The Lab



(b) Room A



(c) Room B

Figure 3: The right shows the resulting values for the target images. Dark indicates a high expected future reward. The left shows the three images with the highest value.

References

[1] R. Basri, E. Rivlin, and I. Shimshoni. Visual homing: Surfing on the epipoles. In *ICCV98*, pages 863–869, 1998.

[2] R. Brooks. Intelligence without reason. In *Proceedings of 12th Int. Joint Conf. on Artificial Intelligence*, 1991.

[3] E. de Jong. The development of a lexicon based on behavior. In Han La Poutr and Jaap van den Herik, editors, *Proceedings of the Tenth Netherlands/Belgium Conference on Artificial Intelligence NAIC'98*, pages 27–36, 1998.

[4] E. de Jong. *Autonomous Formation of Concepts and Communication*. PhD thesis, Vrije Universiteit Brussel, 2000.

[5] M.J. Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 1997.

[6] M. Nicolescu and M.J. Mataric. Learning and interacting in human-robot domains. *IEEE transaction on Systems, Man and Cybernetic*, 2001.

[7] M. Nicolescu and M.J. Mataric. Learning task representations from experienced demonstrations. In *Proceedings, the AAAI Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, 2001.

[8] L. Steels. The origins of intelligence. In *Proceedings on the Carlo Erba foundation, Workshop on Artificial Intelligence*, 1996.

[9] L. Steels. *The Talking Heads Experiment: Volume I. Words and Meanings*. Best of Publishing, Brussels, 1999.

[10] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[11] S.H.G. ten Hagen. Good features to map. In *Proc.IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, USA, October 2003.

[12] S.H.G. ten Hagen and B.J.A. Kröse. Learning to navigate using a lazy map. In *Proceedings of the 11th International Conference on Advanced Robotics*, Coimbra, Portugal, June/July 2003.

[13] P. Vogt. Grounding language about action: mobile robot play follow me game. In *SAB2000 Proceedings Supplement Book*. International Society for Adaptive Behavior, MIT Press, 2000.

[14] P. Vogt. Anchoring symbols to sensorimotor control. In H. Blockeel and M. Denecker, editors, *Proceedings of the 14th Dutch-Belgian Artificial Intelligence Conference, BNAIC'02*, pages 331–338, Leuven, Belgium, October 2002.

Combining Goal Generation and Planning in an Argumentation Framework

Joris Hulstijn[a] Leendert van der Torre[b]

[a] Universiteit Utrecht
[b] CWI Amsterdam

Abstract

In this paper we use formal argumentation theory to study the deliberation cycle of the BOID cognitive agent architecture. We combine goal generation with a planning approach recently proposed by Leila Amgoud. In her formal argumentation framework plans for fulfilling a desire are represented just like arguments that support a conclusion; the framework can thus deal with conflicting desires. In this paper we show how the argumentation approach to planning can be combined with goal generation in the BOID architecture. We also show that combining goal generation and planning leads to interesting new research questions about their relationship.

1 Introduction

An important aspect of intelligent agents is the goal deliberation process, in which an agent decides which goals to pursue. Goal deliberation can be implemented in many ways, though agent architectures often assume separate components for goal generation, goal selection and planning [9, p 76]. Terminology differs; we say that a goal generation process produces potential goals on the basis of an agent's beliefs, desires (internal motivation), and possibly obligations (external motivation). Goal selection is the subsequent process of deciding which consistent subset of goals will be pursued. These selected goals become intentions. Planning is the process of finding a sequence of actions to achieve the intentions.

In this paper we study goal deliberation in the BOID cognitive agent architecture [2, 3, 5]. Goal generation in the BOID architecture has been studied before, but planning is the subject of a joint project with the IRIT laboratory in Toulouse, which is developing an argumentation theoretic account of planning [1].

The BOID architecture uses sets of prioritized default rules to specify the contents of four separate components: Belief, Obligation, Intention and Desire. As in most rule-based systems, heads of rules are compared to the facts, producing a set of applicable rules. Based on a priority order, a rule is selected and applied. Earlier we showed how restrictions on priority orders correspond to agent types [3]. For example, a selfish agent ranks desire rules over obligation rules.

In goal deliberation we have to deal with two kinds of reasoning: forward reasoning from current beliefs to desirable states (deduction), and backward "means-ends" reasoning from desirable states to required actions (abduction). This combination is often problematic. We have observed two problems with goal deliberation in the BOID architecture.

1. The goal generation and goal selection components partly depend on planning, because potential or selected goals are required to be feasible [4, 7]. A goal is feasible when some plan exists that is likely to achieve it. Rao and Georgeff call an agent that only generates feasible goals *strongly realistic*. It is waste of resources to consider infeasible goals. The feasibility restriction requires that planning can provide feedback to goal generation and selection.
2. The application of a priority order is difficult. The easiest solution is to define a local priority order over rules [8]. However, single rules often have undesired consequences. Application of rules should therefore be compared by their outcomes, using a utility value for example. In a possible worlds framework utilities can be expressed, but in a practical agent system this is difficult to achieve. In previous work we have chosen to use extensions, maximally consistent sets of rules, as the main interface between components [2, 3, 5]. The use of extensions has some drawbacks, both practical and conceptual. The extensions themselves become large, and their consistency becomes difficult to check and maintain. The number of extensions multiplies quickly, while the overlap between extensions is considerable. Clearly, we need a clever representation.

To address these problems, we apply techniques from argumentation theory [6]. We combine goal generation with Amgoud's version of Dung's argumentation framework to reason about conflicting desires [1]. The central analogy is the following. A desire or potential goal that has possible 'trees of realization' or plans to achieve it, can be modeled just like an argument which consists of a conclusion with the supporting argumentations. Trees of realization form a declarative representation at the intermediate level between rules and outcomes. The attack relation defined over arguments can serve as a criterium to select goals. Because infeasible goals have no tree of realization, they are excluded from deliberation.

In this paper we use formal argumentation theory to study the deliberation cycle of the BOID architecture. Due to space limitations, we do not discuss the advantages and disadvantages of argumentation theory with respect to theories developed in, for example, conditional logic, logic programming or databases. We just observe that argumentation and dialogue theories have been used to analyze logic, and that defeasible argumentation has been used to study non-monotonic reasoning and reasoning about uncertainty. Theories for conditionals and rules have attracted attention lately with dedicated workshops, and a further unification may be expected during the coming years.

The paper is structured as follows. In section 2 we present goal generation, in section 3 we combine it with Amgoud's argumentation framework, and in section 4 we consider new research questions.

2 BOID goal generation

To keep discussion to a minimum, we only consider the generation of goals from desires, not from obligations and intentions. In the language, we distinguish among proposition variables that we call decision variables and other variables. Decision variables represent atomic actions that need no further plan to be achieved. This distinction is implicit in Amgoud's framework.

Definition 1 (Logic) *Let A be a set of decision variables. Let N be a set of non-decision variables. Let L be a propositional language built from $A \cup N$. Let a literal l of L be a variable or its negation. Let a rule be an ordered nonempty finite list of literals written like $l_1 \wedge l_2 \wedge \ldots \wedge l_{n-1} \rightarrow l_n$. We call $l_1 \wedge l_2 \wedge \ldots \wedge l_{n-1}$ the body of the rule, and l_n the head. If $n = 1$ the body is empty and we write l_n.*

We extend Amgoud's notion of unconditional desires to a set of desire rules D, and we interpret a plan library and background knowledge as sets of rules P and Σ respectively. A desire rule $l_1 \wedge \ldots \wedge l_{n-1} \rightarrow l_n$ in D represents that l_n is desired in the context $l_1 \wedge \ldots \wedge l_{n-1}$, a planning rule in P represents that l_n is achieved if $l_1 \wedge \ldots \wedge l_{n-1}$ is achieved, and a background rule in Σ represents that l_n is true when $l_1 \wedge \ldots \wedge l_{n-1}$ is true. Since a decision variable needs no planning rules, we require that the head of a planning rule is not a decision variable.

Definition 2 (Desire-plan description) *Let A, N and L be as defined in definition 1. A* desire-plan description *is a tuple $\langle D, P, \Sigma \rangle$ with D, P and Σ sets of rules from L, such that the heads of rules in P are built from a variable in N.*

Here are some examples of desire-plan descriptions. The first example extends example 1 of [1]. Note that the examples in this paper are meant to illustrate the definitions. We realize that further validation of the approach is necessary.

Example 1 (Travel) *Assume variables $A = \{ag, fr, hop, dr, sol, w, cp, gp\}$ and $N = \{wca, jca, hjor, dlc, fp, pa, t, vac\}$, with the interpretation:*

wca	:	*there is a war in Central Africa*	*ag*	:	*to go to the agency*
jca	:	*to journey to Central Africa*	*fr*	:	*friend brings the tickets*
hjor	:	*have a job on return*	*hop*	:	*to go to a hospital*
dlc	:	*deadline for submission is close*	*dr*	:	*to go to a doctor*
fp	:	*to finish a paper before going*	*sol*	:	*to solicit for work*
pa	:	*the paper is accepted*	*w*	:	*to work*
t	:	*to get the tickets*	*cp*	:	*call the program chair*
vac	:	*to be vaccinated*	*gp*	:	*write a good paper*

Consider the following desire-plan description:

$$D = \{\neg wca \rightarrow jca,\ jca \rightarrow hjor,\ dlc \rightarrow fp,\ fp \rightarrow pa\}$$
$$P = \{t \wedge vac \rightarrow jca,\ ag \rightarrow t,\ fr \rightarrow t,\ hop \rightarrow vac,\ dr \rightarrow vac,\ sol \rightarrow hjor,\ w \rightarrow fp,\ cp \rightarrow pa,\ gp \rightarrow pa\}$$
$$\Sigma = \{\neg wca,\ dlc,\ w \rightarrow \neg ag,\ w \rightarrow \neg dr,\ w \rightarrow \neg hop\}$$

In some contexts, the agent has desires to travel, have a job, to finish a paper and get this paper accepted. There are several ways to achieve these desires. For example, to get a paper accepted this agent can either write a good paper or call the program chair.

The second example is concerned with food and wine.

Example 2 (Dinner) *Let $A = \{e, r, d, t, w\}$, $N = \emptyset$ and represent dinner options with e for entrecote, r for red whine, d for daurade, t for trout and w for white wine. Let dinner preferences be $D = \{e,\ e \rightarrow r,\ d,\ t,\ d \wedge t \rightarrow w\}$, $P = \emptyset$ and $\Sigma = \emptyset$. The agent prefers to have red wine with red meat, and white wine with fish.*

A goal set is a set of related desires. First we define what it means to apply desire rules to generate a goal set, which is analogous to applying inference rules in classical logic. Then we define a goal set to be a set of heads of desire rules that cannot be further split up into separate goal sets. This represents the fact that desires in a goal set are related. The definition may seem circular, but note that the size of the goal sets shrinks and that they are finite. The notion is thus well-defined. Finally we define maximal goal sets with respect to set inclusion.

Definition 3 (Goal set) *The closure of a set of rules R on a set of literals S, written as $C(R, S)$, is defined by $C(R, S) = \bigcup_{i=0}^{i=\infty} S^i$ with $S^0 = S$ and $S^{i+1} = S^i \cup \{l \mid l_1 \wedge \ldots \wedge l_n \rightarrow l \in R$ and $\{l_1, ..., l_n\} \subseteq S^i\}$. We write $H(R)$ for the set of heads of the rules in R. Given a desire-plan description $\langle D, P, \Sigma \rangle$, a finite set of literals GS is a* goal set *iff there exists a subset D' of D such that:*

1. *$GS = C(D' \cup \Sigma, \emptyset) \cap H(D')$;*
2. *$C(D' \cup \Sigma, \emptyset)$ is consistent, i.e., does not contain two literals l and $\neg l$;*
3. *There is no set of goal sets $\{GS_1, \ldots, GS_n\}$ with each $GS_i \neq GS$ and $GS = GS_1 \cup \ldots \cup GS_n$.*

A maximal goal set *is a goal set which is maximal with respect to set inclusion.*

The following two examples illustrate that a goal set is a set of *related* desires.

Example 3 (Travel, continued) *The goal sets are $\{jca\}$, $\{jca, hjor\}$, $\{fp\}$ and $\{fp, pa\}$. The set of desires with {jca, hjor, fp, pa} is not a goal set, because it can be split in $\{jca, hjor\}$ and $\{fp, pa\}$. The latter two are the maximal goal sets. Here, jca and hjor are related, because the desire to have a job on return from travel (hjor) is conditional on making a journey to Central Africa (jca). Likewise, (fp) and (pa) are related, because the desire to have a paper accepted (pa) is conditional on finishing a paper before going to Central Africa (fp).*

Example 4 (Dinner, continued) *We have the following goal sets: $\{e\}$, $\{e, r\}$, $\{d\}$, $\{t\}$, $\{d, t, w\}$. The set of desires $\{e, r, d, t, w\}$ is not a goal set, because it can be split in $\{e, r\}$ and $\{d, t, w\}$. This expresses for example that the desires for e and r are related, but that desires for e and d are not related. The sets $\{e, r\}$ and $\{d, t, w\}$ happen to be the maximal goal sets. Here, e and r are related, because the desire for red wine (r) is conditional on having entrecote (e). Likewise, the desire for white wine (w) is conditional on having daurade (d) and trout (t) for dinner.*

3 Amgoud's framework for planning

In this paper we combine Amgoud's definitions of an argumentation framework with our goal sets. Her motivation to study plan generation in an argumentation framework is to deal with conflicting desires. She also shows how plan argumentation differs from belief argumentation. Actions and sub-actions are defined as tuples $\langle h, H\rangle$ analogous to claims for h with support H [1, 6].

Definition 4 (Action) *An* action *for a desire-plan description $\langle D, P, \Sigma\rangle$ is:*

- $\langle h, \emptyset\rangle$ *for any decision variable h, called an atomic action; or*
- $\langle h, \{l_1, \ldots, l_n\}\rangle$ *for any rule $l_1 \wedge \ldots \wedge l_n \rightarrow h \in P \cup \Sigma$.*

Two actions $\langle h_1, H_1\rangle$ and $\langle h_2, H_2\rangle$ conflict iff $C(\Sigma, \{h_1, h_2\} \cup H_1 \cup H_2) \vdash \bot$.

From these actions Amgoud constructs so-called realization trees, which serve as a way to represent and reason about plans. Each node of the tree is an action and each child of an action is one its subactions. We extend Amgoud's definition of a realization tree for goal sets, in which the root contains a set of desires. Realization trees are plans to achieve a particular goal set, and may thus also be called goal achievement trees.

Definition 5 *A* tree of realization *g for a goal set GS, written as $g(GS)$, is a finite tree consisting of actions such that*

- $\langle \top, GS\rangle$ *is the root of the tree;*
- *A node $\langle h, \{l_1, \ldots, l_n\}\rangle$ has exactly n children $\langle l_1, H_1\rangle$, ..., $\langle l_n, H_n\rangle$;*
- *The leaves of the tree are atomic actions.*

Consider the holiday plans of example 1. Figure 1 visualizes two realization trees. One for the goal set $\{jca, hjor\}$, which is based on desire rules $\neg wca \rightarrow jca$ and $jca \rightarrow hjor$, and one for $\{pa, fp\}$, based on desire rules $dlc \rightarrow fp$ and $fp \rightarrow pa$. Note that the goal generation steps are not visualized.

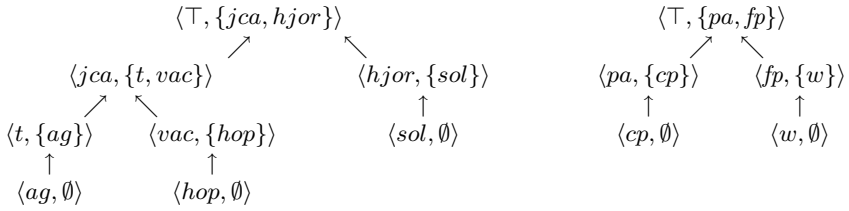


Figure 1: Two trees of realization

Just like arguments, trees of realization may conflict. Therefore it makes sense to use concepts from argumentation theory. In general, an argumentation framework is defined as a set of arguments with a binary relation that represents which arguments attack which other arguments [6]. Here, trees of realization play the role of arguments; the attack relation is derived from conflicts between actions. For more details on this argumentation framework, we refer to Amgoud's paper [1].

Definition 6 *A* system handling conflicting desires *(*SHD*) is a tuple* $\langle G, Attack\rangle$ *such that* G *is a set of realization trees and* $Attack$ *is a binary relation over* G*.*

Based on the notion of conflict between actions (definition 4), we can specify a particular attack relation. The argumentation notions of *defence*, *preferred extension* and *basic extension* are defined accordingly. Note that Amgoud's definition of defence differs from the usual one.

Definition 7 *Let* $\langle G, Attack\rangle$ *be an* SHD *such that* G *contains all realization trees that can be derived from goal sets of a given desire-plan description* $\langle D, P, \Sigma\rangle$*. Let* $S \subseteq G$ *and* $g, g_1, g_2 \in G$ *be (sets of) realization trees.*

- $\langle g_1, g_2\rangle \in Attack$*, i.e.,* g_1 *attacks* g_2*, iff there exist actions* a_1 *and* a_2 *in the nodes of* g_1 *and* g_2 *respectively, such that* a_1 *and* a_2 *conflict.*
- S *is attack free iff there are no* $g_1, g_2 \in S$ *such that* g_1 *attacks* g_2*.*
- S *defends* g *iff for all* $g_1 \in G$ *such that* g_1 *attacks* g*, there is an alternative* $g_2 \in S$ *with* $root(g_1) = \langle \top, GS\rangle$ *and* $root(g_2) = \langle \top, GS\rangle$*, for some* GS*.*
- S *is a preferred extension iff* S *is maximal w.r.t. set inclusion among the subsets of* G *that are attack free and that defend all their elements.*
- S *is a basic extension iff it is a least fixpoint of the function* $F(S) = \{g | g \text{ is defended by } S\}$*.*

These argumentation notions are illustrated by the travel example.[1]

Example 5 (Travel, continued) *There are 7 realization trees. Two for goal set* $\{jca\}$*, one with hop and one with dr, and therefore also two for* $\{jca, hjor\}$*. There is one tree for* $\{fp\}$ *and again two for* $\{fp, pa\}$*, one with cp and one with gp. The two trees in figure 1 attack each other, amongst other reasons because actions* $\langle ag, \emptyset\rangle$ *and* $\langle w, \emptyset\rangle$ *conflict. In fact any tree for* $\{jca, hjor\}$ *or* $\{jca\}$ *attacks any tree for* $\{fp, pa\}$ *or* $\{fp\}$*. These two clusters of trees are internally attack free. There is no non-trivial defend relation in this example. The preferred extensions are the set of trees for the cluster* $\{jca, hjor\}$ *or* $\{jca\}$*, and the set of trees for the cluster* $\{fp, pa\}$ *or* $\{fp\}$*. The basic extensions are the same.*

4 Intertwining goal generation and planning

Thus far, the generation of goal sets and of realization trees (planning) have been largely kept separate. However, we can extend realization trees (plans) with a representation of goal generation steps. To distinguish goal generation steps from planning steps, we let the arrow point in the opposite direction. Consequently the representation is no longer a tree and will from now on be called a *directed acyclic graph* or DAG. The arrows indicate the flow of reasoning in the following sense. A downward arrow for goal generation represents a deduction step. In example 1, from the desire jca and $jca \rightarrow hjor$ we may conclude $\{jca, hjor\}$. An upward arrow indicates an abduction step. For example, from jca we can abduce $\{t, vac\}$ with the rule $t \wedge vac \rightarrow jca$. The following example illustrates that in some cases, goal generation and planning are even more intertwined.

[1] The definitions and examples are implemented in Prolog, see `http://boid.info/boidarg/`.

Example 6 *Let $A = \{a, b\}$ and $N = \{p, q\}$. Consider the desire-plan description $D = \{\top \rightarrow p, a \rightarrow q\}$, $P = \{a \rightarrow p, b \rightarrow q\}$, $\Sigma = \emptyset$. An example of a desire for q triggered by a, is the desire to drink during extended physical exercise, such as a rowing race. If we extend the formalism to obligations (external motivation), another example would be the obligation to pay on the metro. The obligation is triggered by the traveling action; it is not a precondition in the usual sense. In this framework, preconditions or required resources are best represented by a planning rule. The only goal set is $\{p\}$, and the only tree of realization for p contains the actions $\langle p, \{a\}\rangle$ and $\langle a, \emptyset\rangle$. However, intuitively the action $\langle a, \emptyset\rangle$ seems to trigger a further desire for q. When we extend a tree of realization with goal set generation, we get the* DAG *on the left of figure 2.*

Example 7 *Let A and N be as in example 6. Consider $D = \{\top \rightarrow p, a \rightarrow q\}$, $P = \{a \wedge b \rightarrow p, a \wedge b \rightarrow q\}$, $\Sigma = \emptyset$. This* DAG *is shown at the right of figure 2.*

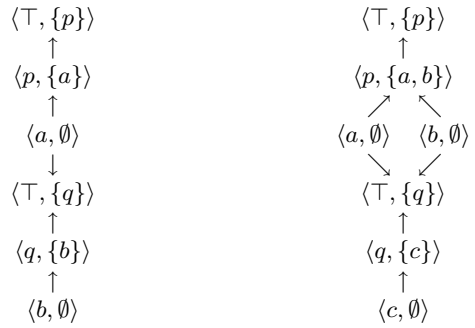


Figure 2: Dags of realization with alternation for examples 6 and 7

5 Concluding remarks

In this paper we have combined BOID goal generation with planning in an argumentation framework. As a result, argumentation notions like attack, defend, maximally preferred extensions and basic extensions, can be used in a model of goal deliberation. Reconsider the two problems discussed in the introduction.

1. The preferred extension corresponds to the maximal set of mutually consistent feasible goals: for each goal, there is a plan and for all possible conflicts between plans, there is an alternative. Similarly, the basic extension can be interpreted as a stable set (self-defending) of feasible goals. Therefore such goal sets are good candidates to become intentions.
2. We suggest to use argumentation extensions, which are sets of realization trees. Realization trees provide an intermediate structure to allow for consistency checks and comparison. Goals without a defended realization tree can be filtered out.

The use of realization trees thus promises to provide a representation layer at a level between individual rules and complete outcomes. Because an infeasible goal does not have a defended tree of realization, the feasibility restriction is easily applied.

Dags of realization promise a tighter connection between goal generation and planning. In particular, they are able to express alternating goal generation and planning steps, illustrated by example 6. This possibility illustrates that more research on the nature of the relationship between goal generation and planning is required. In future work we will consider ways of extending the argumentation notions to dags of realization. Dags of realization form a representation that addresses feedback between planning and goal generation. However, such a representation format only provides a first step; more cognitive and applied research into the nature and desirability of feedback between goal generation and planning remains necessary.

Acknowledgements

Thanks to Leila Amgoud for discussions on the topics raised in this paper when the second author visited the IRIT laboratory in March 2003. Thanks also to Mehdi Dastani for valuable input.

References

[1] L. Amgoud. A formal framework for handling conflicting desires. In *Proceedings of Seventh European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2003)*, LNCS 2711, 552-563, Springer Verlag, 2003.

[2] J. Broersen, M. Dastani, Z. Huang, J. Hulstijn, and L. van der Torre. The BOID architecture. In *Proceedings of the Fifth International Conference on Autonomous Agents (AA'2001)*, 9-16, ACM Press, 2001.

[3] J. Broersen, M. Dastani, J. Hulstijn, and L. Van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):431–450, 2002.

[4] P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[5] M. Dastani and L. van der Torre. What is a normative goal? Towards goal-based normative agent architectures. In *Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA'02)*. Springer, to appear.

[6] P. M. Dung. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[7] A. S. Rao and M. P. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–343, June 1998.

[8] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[9] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2002.

A Symbolic Approach to Music Recognition

Nico Jacobs Filip Van den Borre Lennert Smeets
Evarest Schoofs Hendrik Blockeel

Computer Science Department, KULeuven
Celestijnenlaan 200A, B-3001 Leuven

Abstract

Recognising a piece of music from a user whistling a small part of it is not an easy task: for instance users switch to higher or lower octaves and stretch or crop the notes. In this work we propose a symbolic technique to recognise monophonic music from whistling: we first record the whistling and store this as a sequence of tone frequencies and tone durations, then we compare different approaches to normalise this data. After normalisation we experiment with variants of the edit distance and we compare this approach with n-grams.

1 From Whistling to Tones

Digitised music comes in two important categories. One is to sample the sound signal during a performance, comparable to the recording of cassette tapes or vinyl disks. Typical examples of this are the wave format and the mpeg 1 layer III (mp3) format. Playing this type of records is easy: the samples contain enough information to replicate the original signal. The disadvantages of these types of recording are that they require a lot of storage space (about 10 megabyte per minute for typical waves and one megabyte per minute for typical mp3 recordings) and the information in the recording cannot be interpreted in the sense that you can distinguish separate notes or instruments. The other approach is to store symbolic information: a script which tells which instrument should play a specific note at a specific moment. To play such a record a script is executed in chronological order and the music is *generated*. These types of digital music are best compared with music scores (paper music); MIDI and ABC are the most common used formats. The advantages of these symbolic approaches is that the music is interpretable and file sizes are small (about 10 kilobyte per minute, 1% of the size of an mp3 file). Disadvantages are that it is up till now impossible to transform polyphonic wave recordings automatically into symbolic recordings and that the quality of the produced music depends not only on the recording but also on the generator.

In this paper we define recognising music as retrieving the correct music record from a database when an approximation of a small piece of that record was given as a query. We focus on techniques that use a symbolic representation. First we have to be able to represent a database of songs and the query in a symbolic representation. MIDI is the most common symbolic representation, but it contains a lot of information that is irrelevant for elementary music recognition: the instrument

that plays specific notes, specific ways in which a note should be played et cetera. We will use a restricted representation: a sequence of notes, where each note is only represented by its frequency (in Hertz) and its duration (in milliseconds). It is easy to automatically generate such sequences from monophonic MIDI files, for polyphonic MIDI files we require the user to identify the track that contains the main melody. Transforming the whistling to this format is a bit more complex: we obtain frequency samples directly from the microphone. Via fast Fourier transformation we identify for each sample the most important frequency and we remove all frequencies except the one with the highest energy. We split up this stream into blocks whenever either a silence or an abrupt change of frequency occurs. We consider each such block as one note with a duration equal to the block length. As frequency our current implementation takes the frequency at the start of the block. Further research should investigate the consequence of this decision: maybe taking the average or median value results in a better representation of the intended note. During experiment we also note that people 'correct' themselves towards the end of a note, maybe taking a weighted average with more weight on the end of the note could be useful?

For the remaining of the paper we will work with a set of 65 whistling samples obtained from about 20 people of which nearly all had no musical training. We work on a very restricted dataset because we can only handle monophonic MIDI files. Since all files we found on the Internet were polyphonic we had to create the database files ourselves and so we restrict to a database with only 21 different entries. This is a mixture of children songs ('London Bridge', 'Kortjakje', ...) and popular songs (songs from Abba, Eric Clapton, ...). In section 2 we discuss the need for normalisation of the data and propose different approaches to this. Then we investigate different approaches to calculate a distance between normalised sequences in section 3. We empirically evaluate these proposals in section 4 before we conclude.

2 Data Normalisation

Humans don't whistle a piece of music at exactly the same frequency and duration as it is stored in the database. One problem is that most users pick more or less at random a frequency to start whistling and whistle the next notes relative to this starting note. Something similar occurs with the duration of the notes: they whistle a piece of music slower or faster than the original. Users also tend to shift one or more octaves up or down as soon as they notice they reach the limits of what they can whistle. We try to alleviate these problems as much as possible in the normalisation phase. Another problem is that users add notes ('filler notes') or leave out notes because either they forgot them or it makes the piece easier to whistle. We will try to cope with this second type of problems in section 3.

To solve the first type of problems we test four normalisations on the input. Since our input consists of two sequences (a sequence of frequencies and a sequence of durations) we can transform each of them separately. In this discussion we simply consider the input to be a sequence v of real values $v_1, v_2, \ldots, v_n$. We

briefly discuss these transformations and the motivation for testing them.

1. **standard normalisation**: $v_i^{norm} = (v_i - min(v))/(max(v) - min(v))$. By mapping the sequences to a range $[0, 1]$ it makes them easier to compare. However, this technique is very sensitive to outliers and doesn't take into account the fact that a linear increase in tone causes an exponential increase in frequency (and similar for duration).

2. **logarithmic averaging**: $v_i^{avg} = log(v_i)/avg(log(v))$. We first take the logarithm of every value in the sequence and then divide each element by the average. Taking the average is less sensitive to outliers than the previous approach and the logarithm helps solving the problem with the exponential scale.

3. **relative representation**: $v_i^{rel} = log(v_i) - log(v_{i-1})$. People with no musical training will most often remember and reproduce music in a relative way (e.g. 'the second note is two tones higher than the first' instead of 'first a re, then a fa'). Note that this transformation shortens the sequence one element: v_1^{rel} does not exist.

4. **sound contour**: $v_i^{contour} = \nearrow$ if $v_i > v_{i-1}$, $\searrow$ if $v_i < v_{i-1}$ and $=$ if $v_i = v_{i-1}$. This is a more abstract version of the previous transformation: we only store the fact that the value increases, decreases or stays the same [3]. Since it is also a relative representation, it also shortens the sequence one element.

Another problem that we can solve in the normalisation phase is that of discretisation. Some of the techniques in section 3 require the data to be nominal, therefore we need to split up the real values that we obtain from the first three transformations in ranges so that we can map each range onto a nominal value. This is an unsupervised discretisation task for which different approaches exist. In this paper we use a simple approach: we divide the range of the values into n equal width bins [1]. We test two possible values for n, namely 10 (10-bin) and the logarithm of the number of unique values in the transformed data (log-bin).

3 Distances Between Songs

To obtain the song in our database that best matches a whistled 'query' of the user, we define a distance between the normalised query and the normalised data in the database. For this task we will need an approximating substring matching technique. There are two such techniques that are widely used: n-grams and the edit distance.

3.1 N-grams

For a fixed $n \in \mathbb{N}$ (typical values for n are 3 or 4, for $n = 1$ we obtain a bags of words representation) and two given sequences $s = s_1, s_2, \ldots, s_{l1}$ and $t = t_1, t_2, \ldots, t_{l2}$

we define an n-gram distance between s and t as $\sum_{i \in sub}(count(i,s) - count(i,t))$ where sub is the set of all substrings of length n of s and $count(i,s)$ is the number of times that substring i occurs in s. This distance only makes sense on nominal data, so we will make use of discretised data

3.2 Edit Distance

The edit distance [5] between two strings s and t is the minimal number of insert, delete and replace operations to transform s into t. It is usually defined incrementally: ($s_{1,i}$ denotes the prefix of length i of sequence s)

$$d(s_{1,i}, t_{1,j}) = min \begin{cases} d(s_{1,i-1}, t_{1,j}) + w_{indel} \\ d(s_{1,i}, t_{1,j-1}) + w_{indel} \\ d(s_{1,i-1}, t_{1,j-1}) + \begin{cases} w_{replace} \text{ if } s_i \neq t_j \\ 0 \text{ if } s_i = t_j \end{cases} \end{cases}$$

w_{indel} is the cost of an insert or delete operation[1] and $w_{replace}$ is the cost of replacing an element. This distance can be calculated in $O(mn)$ time and $O(min(n,m))$ space, where m and n are the length of the first and second sequence.

When we use this to find the best match between a music fragment and a complete song we face two problems. One problem is that comparing real values (for the 'replace' action) will always yield a cost increase of $w_{replace}$. This problem can be solved in three ways: we can either discretise the data, we can consider two real values different only if they differ more than a certain threshold (i.e. replacing $s_i = t_j$ by $|s_i - t_j| < threshold$) or we can define a continuous $w_{replace} = f(s_i, t_j)$. The second problem is that there are many delete operations necessary to transform a long song sequence into a short fragment sequence. This will introduce the problem that longer songs will have a higher distance to a fragment, even when they match the fragment completely. This problem is solved by using a variant of the edit distance that does not take delete costs at the start or the end of the longer sequence into account [2].

4 Experiments

We test how well we can retrieve the correct song from only the frequency or the duration information. We test this with each normalisation, discretisation and approximate string matching algorithm discussed in the previous sections. Recall that there are 65 queries on a database with 21 entries.

We use two criteria to measure the success of a classification. Our first criterion, precision, is the number of times the classifier predicts the correct song, and is an important criterion if you only provide one prediction to the user. When you provide the user with a ranking of the songs in order of decreasing likelihood of being the requested song, another criterion becomes important: how far down this

[1] these have equal cost because inserting an element when transforming s into t is the same as deleting an element when transforming t into s: this guarantees that the distance is symmetric; in the remainder of this text we will refer to this as the *shift cost*

list is the actual song? The average rank of the correct song becomes our second criterion.

4.1 N-grams

The n-gram algorithm only has one parameter, namely the length of the substring. We test the n-gram approach in which we vary the value of n between 2 and 9. We do this for all discretised normalisation methods. When we compare the different discretisation methods we see that log-bin, which creates fewer discretisation bins than 10-bin, performs better. When we look at the different normalisations we see that standard normalisation performs bad and the relative representation works better than the other methods. The contour approach performs good as well, despite its simplistic discretisation. The optimal value of n depends upon the normalisation used. Four and five are on average the best values for n. Notice that even the best approach (5-gram with the log-bin discretisation approach and the relative log representation) only get 15 out of the 65 queries (23.1%) correct, which is much better than random guessing (4.76%) but still very low for practical use, especially since we only have few songs in our database.

When we repeat this experiment with the note duration sequence, we see no structure in the results: both discretisation methods and all four normalisations behave in a similar manner: they fluctuate around seven correct predictions, without any being clearly better or worse than the others. This means that predicting the correct song from a query containing only durations is much harder with n-grams than from a query containing only frequencies. But predicting from a duration sequence still performs significantly better than random guessing, which means that the durations do provide useful information.

The ranking criterion for classification evaluation does not seem to distinguish between any of the n-gram approaches: all values fluctuate around an average ranking of 8.5, which is too high to be practically useful.

4.2 Edit Distance

The edit distance uses two types of costs: insert/delete costs — which in these experiments are independent of the frequency or duration we insert or delete — and a replace cost. As we already mention in section 3.2 there are three variants to deal with the replacements in this continuous domain: either we use discretised data, we use 'regions' around the continuous values to decide whether they are equal or not, or we define the replace cost in function of the difference between the two compared values.

First we use the discretised data which we also used for the n-grams. The precision criterion on frequency data results are comparable with the n-gram results: the relative log representation combined with the 10-bin discretisation and with the relative logarithmic normalisation performs best, a bit above the level of the n-grams (27.7% instead of 23.1%) but this difference is not statistically significant. We use values between one and five for the shift and replace cost; best results are obtained when the shift cost equals the replace cost. When we compare both

using the ranking criterion we see that the average rank is only 7 instead of 8.5. This result is obtained with a high shift cost (five) and a low replace cost (one): apparently optimal settings for the precision criterion are not optimal settings for the ranking criterion.

Second we use the region version of the edit distance. Our criterion to test equality between two reals r_1 and r_2 is as follows: they are considered equal if $(r_1 * (1 - margin)) \leq r_2 \leq (r_1 * (1 + margin))$. We test various settings for this margin as well as different shift and replacement costs and the different normalisations. There are some notable differences with the discretised approach. Logarithmic averaging performs better than the relative log normalisation for both criteria. For both criteria a shift cost of five and a replace cost of one performed best. The results on both criteria depend strongly on the margin used. The results for the logarithmic averaging are shown in figure 1. We see the best performance for the frequency sequence based on the is obtained with a margin of 2% or 3%. For these values this approach is better than the n-grams and the discretised data edit distance approach with respect to the ranking criterion (minimal rank 5.18 versus 6.88 for the discretised edit distance); for the precision criterion it performs as well as the n-grams.

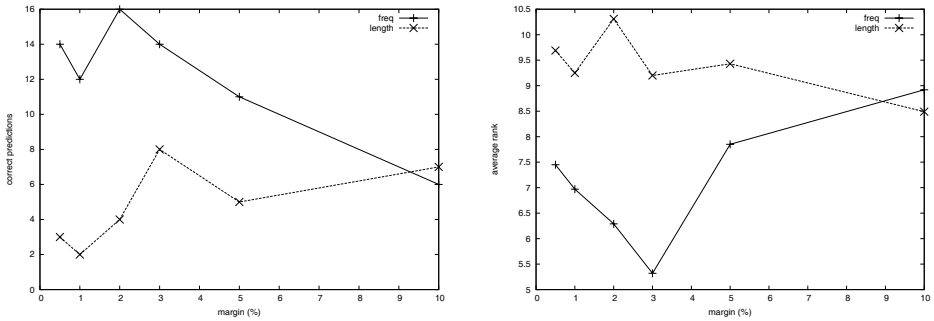


Figure 1: Effect of margin on classification with edit distance using shift cost of five and a replace cost of one. On the left we use the precision criterion, on the right the ranking criterion.

Finally instead of defining the edit distance in terms of matching and non-matching entries we can simply define the continuous edit distance (we restrict ourselves to the absolute difference between the two values as $w_{replace}$, other functions can be used as well)

$$d_c(s_{1,i}, t_{1,j}) = min \begin{cases} d_c(s_{1,i-1}, t_{1,j}) + w_{indel} \\ d_c(s_{1,i}, t_{1,j-1}) + w_{indel} \\ d_c(s_{1,i-1}, t_{1,j-1}) + |s_i - t_j| \end{cases}$$

This variant of the edit distance outperforms the classic edit distance on this problem for both criteria. In figure 2 we compare the normal edit distance approach (`ed` in the graph) with the continuous edit distance (`ced` in the graph). For both of these techniques we calculate how often they correctly predict a song when

they are allowed to make n predictions. The starting point and end point of these graphs are fixed: when we predict nothing we can't predict anything right, and when we are allowed 21 predictions we can predict everything right (remember there were only 21 songs in the database). The straight line between these two points represents the number of correct predictions if we guess randomly. The predictions with the normal edit distance on the duration information results in predictions that are just slightly better than random guessing. When we use the continuous edit distance on this data we get an improvement, but it is clearly worse than when we use the frequency information. Especially when n is small (less than ten) the continuous edit distance on the frequency data outperforms the normal edit distance approach. For $n = 1$ it already obtains 55.4% accuracy, with $n = 5$ the accuracy increased to 73.9% while for the normal edit distance this is 23.1% and 56.9%.

The above results for the continuous edit distance are obtained with the relative log representation. With other normalisations the results are in between the results from the classic edit distance approaches and the results mentioned in the previous paragraph.

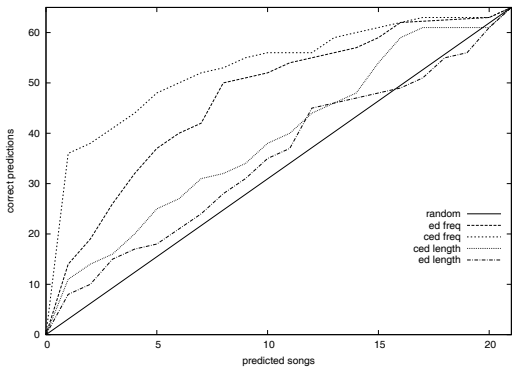


Figure 2: Number of correctly predicted songs in function of the number of songs in a prediction.

In the above experiments we analysed the frequency and duration in isolation. When we combine them by making the replacement cost a weighted sum of the replacement cost of the frequency and the replacement cost of the duration the result is always worse than the continuous edit distance on frequency data alone. The length of the query can also influence the result: maybe queries that are 'too long' result in worse matching? The queries are on average 11.3 notes long, with the shortest only 4 notes and the longest 28 notes. We truncate queries at varying length between 2 and 28 and investigate the effect on both criteria. Up till 15 both criteria improve when we use longer queries. From that point on there are no statistically significant changes in the results if we increase the query length.

5 Discussion and Conclusion

Music information retrieval is a relatively new sub-domain of information retrieval. We investigate a symbolic approach where all database entries are written as sequences of tone durations and tone frequencies. As input we let the user whistle their 'query', which is also transformed in a sequence of tone durations and frequencies. Then approximate string matching techniques are used to find the best match in the database. This approach is comparable to the work of Lemström [3] and the experimental commercial system CubyHum [4]. Our work is more restricted since we use simpler normalisations and distances. However we do not present a single approach but show the effects of different normalisations and approximate string matching techniques on a real world dataset.

Our main conclusion is that the classic edit distance and n-grams both perform badly. This seems to be due to the fact that they discretise the data (items are either equal or different). We obtain much better results with a variant of the edit distance in which we use the difference between two values as replacement cost. Further we illustrate the need and effect of normalisation. Relatively short queries of about 15 notes seem to give the best performance: longer notes don't increase the quality of the prediction but require extra computation time.

The data depends not only on the person that whistles the query, but also on the microphone and sound card used to record the query, the echo in the room et cetera. This makes it difficult to compare different approaches: there is a need for standard benchmarking data sets in this domain. Further research is needed as well, especially regarding the scalability of these approaches. More music-specific preprocessing and normalisation approaches should be tested.

Acknowledgements: The authors thank Quentin Philippe for the implementation of the system and the discussions on the experiments.

References

[1] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Proc. 12th International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, 1995.

[2] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, 1997.

[3] K. Lemström. *String Matching Techniques for Music Retrieval.* PhD thesis, University of Helsinki, Faculty of Science, Department of Computer Science, 2000.

[4] Steffen Pauws. CubyHum: A fully operational query by humming system. In *Proceedings of the third international conference on music information retrieval*, Oct 2002.

[5] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, January 1974.

Preprocessing Documents to Answer Dutch Questions

Valentin Jijkoun Gilad Mishne Maarten de Rijke

Language and Inference Technology Group, ILLC, U. Amsterdam

Abstract

We describe a framework for offline extraction of certain types of information from a document collection, and discuss its usage for answering factoid questions. We implemented this approach as a part of the Dutch Question Answering System developed at the University of Amsterdam. The evaluation of the system using data from the CLEF 2003 Question Answering track shows that our strategy yields a significant improvement in the performance of our overall system.

1 Introduction

With recent advances in computer and Internet technology, people have access to more information than ever before. Much of the information is available in free text with little or no metadata, and there is a tremendous need for tools to help organize, classify, and store the information, and to allow better access to the stored information. Current information retrieval systems allow us to locate documents that might contain the pertinent information, but most of them leave it to the user to extract the useful information from a ranked list. This leaves the (often unwilling) user with a relatively large amount of text to consume.

To address these issues, a number of recent initiatives are aimed at providing highly focused information 'pinpointing.' For instance, in the TREC question answering (QA) track [9] participants are given a large document set and a set of questions; for each question, the system has to return an exact answer to the question and a document that supports that answer.

QA has recently received attention from the information retrieval, information extraction, machine learning, and natural language processing communities [3, 6, 10]. But the field itself is not new; in an overview paper from the mid 1960s, as many as 15 implemented and working systems for question answering are described [8]. These early QA systems, however, were usually natural language front-ends to highly structured data sources, whereas modern systems aimed at addressing TREC-style QA operate on unstructured data (typically, collections of newspaper articles). In this paper we report on preliminary experiments in which we attempt to bring those two traditions together. Our motivation for this work is three-fold. First, while information retrieval techniques are relatively successful at providing near-instant access to the vast amount of data on the web, the precision required of QA systems makes on-the-fly question answering from unstructured

data sources too slow and impractical. Second, for many types of factoid questions used for evaluation purposes, the semantic information that (likely) answers these questions, occurs in very fixed patterns. For example, for questions like "Waar ligt Basra?"(that we classify as a LOCATION question), typical answer patterns are "Basra, slechts vier kilometer van de grens met Iran" and "Basra, in het zuiden van Irak." Our strategy is to exploit such regularities for offline extraction of semantic data so as to make the data available for rapid and easy access. Third, having access to such extracted data will allow us to more easily answer certain types of questions than we would be able to if we only did on-the-fly processing; examples include list questions such as "Noem Europese staatshoofden."

The remainder of this paper is structured as follows. In Section 2 we describe the experimental setting in which we evaluated our ideas, the CLEF 2003 Dutch Question Answering task. In Section 3 we describe our system architecture and contrast it with the canonical QA system architecture. Then, in Section 4 we provide details on the creation and use of various kinds of offline tabular data within the QA scenario; we also assess its effectiveness. In our final section (Section 5) we formulate conclusions and discuss future work.

2 Experimental Setting

This year, the Cross-Language Evaluation Forum (CLEF [2]), a forum dedicated to the development of information retrieval systems for European languages, featured a Question Answering track for the first time; the languages evaluated were Italian, Spanish and Dutch. For the Dutch systems, the corpus was composed of newspaper articles from 1994–1995, taken from the Dutch daily newspapers Algemeen Dagblad and NRC Handelsblad. The total corpus size was about 500MB (72 million words). The question set included 200 factoid question, out of which 10% had no known answer in the corpus.

At CLEF, systems were allowed to return three ranked answers for each question; an answer can either be a 50-byte string which contained the answer, or the exact answer phrase. Each answer is required to be accompanied by *justification*: an identifier for the document from which the answer originated. The University of Amsterdam only submitted runs with exact answers. The CLEF evaluation uses the standard MRR (mean reciprocal rank) scoring metric; however, since the official CLEF assesments have not yet been delivered at this time, we will use a simpler measure in this paper: the percentage of questions which had a correct answer in one of the three answer candidates provided by the system.

3 System Architecture

The general architecture of a QA system, shared by many systems, can be summed up as follows. A question is first associated with a *question type*, out of a predefined set such as DATE-OF-BIRTH or CURRENCY. Then, a query is formulated for the question's expected answer, and issued to an information retrieval engine, which then returns documents that are likely to contain the answer. Those documents

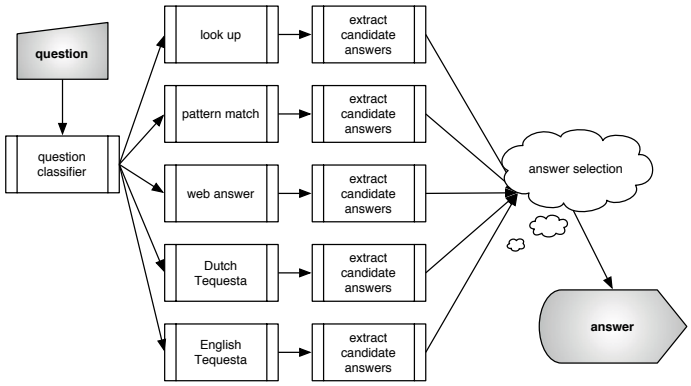


Figure 1: The University of Amsterdam's Dutch Question Answering System.

are sent to an *answer extraction* module, which identifies candidate answers, ranks them, and selects the final answer. On top of this basic architecture, numerous add-ons have been devised, ranging from logic-based methods to ones that rely heavily on the redundancy of information on the World Wide Web [9].

During the design of our QA system, it became evident that there are a number of distinct approaches for the task; some are beneficial for all question types, and others only to a subset. It was therefore decided to implement a *multi-stream* system: a system that includes a number of separate and independent subsystems, each of which is a complete standalone QA system that produces ranked answers; the system's answer is then taken from the combined pool of candidates.

Scientifically, it is interesting to understand the performance of each stream on specific question types and in general. On the practical side, our multi-stream architecture allows us to modify and test a stream without affecting the rest of the system. A general overview of our system is given in Figure 1. The system consists of 5 separate QA streams and a final answer selection module that combines the results of all streams and produces the final answers.

Question Answering Streams. We now provide a brief description of the five streams of our QA system: Table Lookup, Pattern Match, English Tequesta, Dutch Tequesta, and Web Answer.

The *Table Lookup* stream is the focus of this paper; see Section 4 for details. It involves construction of specialized knowledge bases from the collection by preprocessing it. When a question type is determined to be one of the pre-defined types that have a possible answer in these tables, lookup is performed in the respective knowledge base and answers which are found there are assigned high confidence.

In the *Pattern Match* stream, zero or more perl regular patterns are generated for each question according to its type and structure. These patterns indicate strings which contain the answer with high probability, and are then matched against the entire document collection. Here's a brief example:

Question	*2. In welke stad is het Europese Parlement?*
Generated pattern	`Europese Parlement\s+in\s+(\S+)`
Match	... voor het **Europese Parlement in Straatsburg**, dat ...
Extracted Answer	**Straatsburg**

Naturally, these patterns also match strings which are not answers; we therefore count the number of times each candidate answer string matched, and rely on actual answers to appear more frequently than other strings.

The *English Tequesta* stream translates the questions to English using a free web service (WorldLingo, www.worldlingo.com). The auto-translated questions are then fed to *Tequesta*, an existing QA system for English developed in the University of Amsterdam [4], using the English CLEF corpus, and extended with an Answer Justification module to anchor the answer in the Dutch corpus.

The *Dutch Tequesta* is an adaptation of English Tequesta to Dutch and used as an independent stream, provided with the original Dutch newspaper corpus. The modifications to the original system included replacing (English) language specific components by Dutch counterparts; for instance, we trained TNT [1] to provide us with Part-of-Speech tags using the *Corpus Gesproken Nederlands* [5], and a named entity (NE) tagger for Dutch was also developed locally.

The *Web Answer* stream looks for an answer to a question in the World Wide Web, and then finds justification for this answer in the collection. The question is converted to a web query, by leaving only meaningful keywords and (optionally) using lexical information from EuroWordNet. The query is sent to a web search engine (for the experiments in this paper we used Google); if no relevant Web documents are found, the query is translated to English and sent again. If the query yields some results, words and phrases appearing in the snippets of the top results are considered as possible answers, and ranked according to their relative frequency over all snippets. The Dutch NE tagger and some heuristics were used to enhance the simple counts for the terms (e.g., terms that matched a TIME named entity were given a higher score if the expected answer type was a date). Finally, justifications for the answer candidates are found in the local Dutch corpus.

While each of the above streams is a "small" QA system in itself, many components were shared between the streams, including, for instance, an *Answer Justification* module that tries to ground externally found facts in the Dutch CLEF corpus, and a *Web Ranking* module that uses search engine hit counts to rank the candidate answers from our streams in a uniform way. Our *Question Classifier*, which was a shared component as well, relies on pattern matching, making use of the fact that the vast majority of questions of a certain type are formulated in a few typical ways.

4 A Closer Look at the Table Lookup Stream

Before we report on the impact of tabular data, we take a closer look at how the data was obtained.

Extraction. We hand-crafted a small number of regular expressions able to extract information about country currencies, leaders, roles, capitals, inhabitants, abbreviations, and locations. We chose these categories because likely answers to questions asking for such information tend to occur in a small number of fixed patterns. Furthermore, our main aim was to determine the viability of the idea of using knowledge bases generated from the text collection to answer questions; for this reason we opted to work with a small set of hand-crafted high precision extraction patterns. However, in a later stage of this work we plan to investigate machine learning techniques for automatically extracting patterns; such techniques have been used extensively in the field of information extraction [7].

In Table 1 we list the categories for which we created knowledge bases, plus the number of facts per category. The "Adjective-location" category concerns geographic information of the following type "Bhopal, stad, in India, NH19940125-0036", where the first field indicates a location, the second its type, the third a country or region in which it is located, and the fourth the identifier for the document from which it was extracted. "Locations" contains similar information, but without the type; "Leaders" has information of the following kind "Beieren, minister van milieu, Peter Gauweiler, NH19940217-0003", and "Roles" generalizes this to also include other roles besides government-related ones.

Type	# Facts extracted	# Unique facts extracted
Abbreviations	14575	6095
Adjective-location	2328	957
Capitals	1922	465
Currencies	41	26
Inhabitants	39	38
Leaders	10740	2456
Locations	4931	4202
Roles	9717	8954

Table 1: Facts extracted from the Dutch CLEF 2003 corpus.

Due to space limitations we can't provide details on the extraction process for each of the above classes; we briefly discuss some typical cases. Abbreviation questions include questions asking for an abbreviation and questions asking for an expansion of an abbreviation. To collect abbreviation-expansion pairs we made a single pass through the document collection to identify strings of capitals in brackets; upon finding one we extracted sequences of capitalized non-stopwords preceding it (with about as many words as the length of the capitalized string).

While extracting abbreviation-expansion information requires no background knowledge, the "adjective-location" category does. Using EU-guidelines for translators on official adjective-to-country/location mappings, we extracted phrases such as "Mexicaanse vulkaan Popocatepetl" to produce facts such as "Popocatepetl,vulkaan,in Mexico,NH19940718-0041".

We used more background knowledge to populate the roles table: from the Dutch part of EuroWordNet we compiled a list of about 900 professions; syntactic appositions of the form "`<name>, <clause-involving-a-profession>,` were

then extracted to obtain descriptions of the people identified using <name>. A sample fact extracted in this manner is "Ally Derks, oprichter en directeur van het International Documentary Filmfestival Amsterdam (IDFA), NH19941207-0070".

None of the regular expressions used for extraction were meant to be exhaustive of all possible varieties of patterns in which the information being extracted occurs. They are straightforward, but reasonably high precision implementations meant to extract a large proportion of the patterns in the text. After the initial harvesting step, various cleaning up steps were applied to filter out noise. Again, in future work we plan to use machine learning techniques (trained on a small sample of manually extracted facts), but for the current proof-of-concept stage of our offline extraction for QA work, we simply devised some rules by hand to reduce noise.

Look up. The generated tables are simple text files, rather than structured databases; in addition, the actual "key" for the lookup in them is not directly given, and has to be analyzed from the format of the question. We therefore use the following method for the lookup: after identifying relevant knowledge base files using the question type, we try to locate lines in tables that match the question focus. If these are not found, we look for lines that contain as many terms from the focus as possible, giving priority to capitalized words. If matching lines are found, the candidate answers are extracted and ranked according to their frequencies.

The Impact of Using Tabular Data. In order to determine the effect of using tabular data, we evaluated the performance of our system in three different ways: with all five streams, with all streams except for *Table Lookup* and with *Table Lookup* alone. An official CLEF assessment exists only for the entire five-stream system, for exact answers; the rest of the evaluations were done by us manually, in a compatible way to the CLEF evaluations, but counting inexact answers as correct ones. Like the official CLEF evaluation, we considered a question answered correctly if at least one of the top three answers given by the system was correct. We refer here to the lenient (non-strict) results (although for answers obtained from the *Table Lookup* stream it is guaranteed that the document given as justification indeed contains the answer).

Table 2 lists the number of correctly answered questions for two sets of questions: both the whole set of 200 questions, and the 187 questions that contain answers in the collection. In the first case the performance is somewhat better — mainly because our system always includes NIL (*no answer*) in the top three answers, and thus according to our evaluation scheme, questions without answer in the collection are always answered correctly.

	Only	Without	All five streams	
# Questions	*Table Lookup*	*Table Lookup*	Exact	Inexact
200 (all)	54 (27%)	64 (32%)	89 (45%)	102 (51%)
187 (with answer)	41 (22%)	51 (27%)	76 (41%)	89 (48%)

Table 2: Evaluation: the number of questions answered correctly.

Looking at the bottom row in Table 2, we see that the *Table Lookup* stream

provides correct answers for 22% of the questions, and, more importantly, 14% of the questions was only answered correctly by the *Table Lookup* stream (i.e., not by any of the other four streams).

So where did the *Table Lookup* stream prove to be especially helpful? And how significant was the contribution? Table 3 gives a breakdown in terms of the question categories for which the system with the *Table Lookup* stream gives correct answers, while the system without *Table Lookup* fails.

# Questions	Category	Example
8	Capitals	Wat is de hoofdstad van Zuid-Afrika?
7	Inhabitants	Hoeveel inwoners heeft Sydney?
5	Roles/Leaders	Wie is de voorzitter van de Europese Commissie?
4	Abbreviations	Waar staat GATT voor?
3	Locations	In welke stad is het Europese Parlement?
1	Currencies	Hoe heet de Chinese munteenheid?

Table 3: Categories of the questions for which the *Table Lookup* stream helps.

Summing up the entries in column 1 of Table 3, we see that the *Table Lookup* stream correctly answers 28 questions not answered correctly by the other streams. In total, 72 questions looked for answers that could (in principle) be present in our tables. Table 4 gives the performance analysis for these questions: the number of questions that were correctly vs. incorrectly answered by the *Table Lookup* stream alone and all five streams vs. the four streams without *Table Lookup*.

		Table Lookup		*All five streams*	
		correct	incorrect	correct	incorrect
four streams	correct	13	8	19	2
	incorrect	29	22	28	23

Table 4: Breakdown of *Table Lookup* results.

On the types of questions on which it could potentially make a difference, the *Table Lookup* stream made a significant difference (using the sign test, with $p < 0.01$). It is worth noting that the *Table Lookup* stream still leaves lots of room for improvement: only 42 out of 72 relevant questions were answered correctly by the *Table Lookup* stream (58.3%); note that 6 of the missing answers were found by the other four streams (at the expense of 1 previously correct answer), leading to a total of 47 out 72 relevant questions answered correctly (65.3%).

The most common errors encountered in the incorrect answers produced by the *Table Lookup* stream were:

- Some of the heuristics used for retrieving information from the tables did not always work. For this reason, e.g., for question "Wie is de president van Rusland?" the system answered "Rusland."
- Failure to extract the required information. E.g. the sentence "... Australië's formele staatshoofd, de Britse koningin Elizabeth..." didn't produce the entry (Australië, formele staatshoofd, de Britse koningin Elizabeth) in the Leaders table, because there was no pattern for this type of phrase.

Addressing these errors is part of our ongoing work.

5 Conclusion

In this paper we explored the use of offline generated lookup tables for answering certain types of factoid questions. The idea was implemented in a *Table Lookup* stream as part of our participation in the CLEF Dutch QA evaluation exercise. We found that the *Table Lookup* stream made a significant difference, providing correct answers for 58% of all relevant questions. Our ongoing and future work concerns adapting the ideas described here to the AQUAINT corpus used for the TREC QA evaluation, applying machine learning techniques to identify suitable patterns for populating tables and to clean up the output of those patterns. In addition, we want to extend our patterns to include additional question categories, such as age and date of birth or death.

Acknowledgements

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. Maarten de Rijke was also supported by NWO under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, and 612.000.207.

References

[1] T. Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, 2000.

[2] CLEF: Cross-Language Evaluation Forum. URL: `http://www.clef-campaign.org`.

[3] M. Maybury, editor. *AAAI Spring Symposium on Future Directions in Question Answering*, 2003.

[4] C. Monz and M. de Rijke. Tequesta: The University of Amsterdam's Textual Question-Answering System. In *Notebook papers TREC-10*, 2001.

[5] N. Oostdijk. The Spoken Dutch Corpus: Overview and first evaluation. In *Proceedings LREC 2000*, pages 887–894, 2000.

[6] M. de Rijke and B. Webber, editors. *EACL 2003 Workshop on Natural Language Processing for Question Answering*, 2003.

[7] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings AAAI-96*, pages 1044–1049, 1996.

[8] R. Simmons. Answering English questions by computer: A survey. *Communications of the ACM*, 8:53–70, 1965.

[9] E.M. Voorhees. Overview of the TREC 2002 question answering track. In Voorhees and Harman [10].

[10] E.M. Voorhees and D.K. Harman, editors. *The Eleventh Text REtrieval Conference (TREC 2002)*. National Institute for Standards and Technology, 2003.

Combining Exploration and Reliability in Coevolution

Edwin D. de Jong [a]

[a] Universiteit Utrecht, ICS, DSS Group
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands.
dejong@cs.uu.nl http://www.cs.uu.nl/~dejong

Abstract

Coevolution can in principle circumvent the difficult problem of designing a fitness function; this may remove harmful biases, and thereby improve search performance. Recently, based on Evolutionary Multi-Objective Optimization, the feasibility of accurate evaluation in coevolution has been demonstrated theoretically. A current challenge in coevolution is how this theoretical promise may be translated into practical algorithms. We investigate a setup consisting of multiple subpopulations, each with a varying degree of exploration. It is shown that by allowing these subpopulations to interact, the desirable aims of exploration and reliability can be combined.

1 Introduction

Coevolutionary algorithms [2, 1, 13] can address problems where the quality of an individual is determined by its outcomes in several *tests*. This description defines the broad class of *test-based problems*, ranging from supervised learning (where a learning example is a test) to board games such as chess or Go, where each opponent can be viewed as a test. The number of possible tests in such problems is typically very large, and evaluating an individual on all possible tests is therefore generally infeasible. Most evolutionary algorithms therefore require the user of the algorithm to specify a fitness function to be used for evaluation purposes. However, as the ideal evaluation function for test-based problems is generally unknown, choices in defining such a fitness function can greatly influence the performance of the algorithm employing it.

In coevolution, the problem of choosing a fitness function is avoided; rather than having user-defined evaluations along with the biases these are likely to introduce, coevolutionary algorithms evaluate individuals based on interactions with other evolving individuals. Thus, the examples in supervised learning or the opponents in chess are themselves evolving individuals, and the coevolutionary process decides which tests to use for evaluating other individuals.

While coevolution may avoid certain biases that are due to inaccurately determined fitness functions, an important question is to what extent an evolved set of tests can provide accurate evaluation. Indeed, apart from a number of tantalizing

successful results [13, 16, 15], coevolutionary methods are notorious for their unreliability. A particular problem is that it has long been unclear how the evolving tests should be evaluated; if these are set up in symmetric competition with the learning individuals, there is no guarantee that all underlying *objectives* of the individuals will be tested. As an example, the members of two competing populations of chess players may continue to adapt to individuals in the other population, without making any progress in the long run [5, 18]. This *Red Queen* problem can be understood by recognizing that there may be *multiple* objectives underlying performance in a problem, such as strategy and tactics in chess, or fuel-efficiency, esthetics and costs in car design. If individuals are not tested on *all* underlying objectives of a problem, progress in one objective will often be accompanied by undetected regress for the untested objectives.

Recently, based on Evolutionary Multi-Objective Optimization (EMOO) [9], the feasibility of accurate coevolutionary evaluation has been theoretically confirmed. It was proven that for any set of learners, a small set of tests exists that provides all information relevant to the evaluation of the learners [6, 7, 8]. Moreover, this *Complete Evaluation Set* can be approximated by practical algorithms, as it provides an operational criterion for the evaluation of the *tests*; by evolving tests according to this criterion, approximation of the Complete Evalution Set can be guaranteed in the limit [8]. Another approach to the theoretical study of coevolution is based on order theory; using this formalism, a set related to the Complete Evaluation Set has been described [3], and geometrical aspects of coevolution are investigated [4].

A current challenge in coevolution is how the theoretical promise of accurate evaluation without a user-defined fitness function may be translated into practical algorithms. A first algorithm based on approximating the Complete Evaluation Set is the DELPHI algorithm. By making strict choices regarding the replacement of individuals, this algorithm is able to achieve sustained progress in all underlying objectives for several difficult test problems [7]. In order to apply theoretically justified coevolutionary algorithms in practice, a number of hurdles must be overcome. Perhaps the most important limitation resulting from strict replacement criteria is that it may reduce the potential for *exploration*.

In this paper we focus on this issue by presenting and demonstrating a novel technique that combines exploration and reliability. The technique employs a number of *subpopulations*. Inspired by the method of parallel tempering [12], each subpopulation has a different *temperature*, which controls its degree of exploration. Since the parents used for crossover and mutation can come from any subpopulation, genetic material can spread accross the subpopulations. In this way, a 'cold' subpopulation can reach improved locations in the search space. Moreover, while the highly explorative 'hot' subpopulations would normally be likely to lose high quality individuals, the cold subpopulations function as an archive that allows them to rediscover such individuals. It is found that by this system of interacting subpopulations, the desirable features of exploration and stability can be successfully combined.

2 Coevolution: the DELPHI Algorithm

The DELPHI algorithm distinguishes between two types of individuals: *learners* are candidate solutions whose performance is to be optimized, while the *evaluators* are tests that are used to evaluate the learners. The DELPHI algorithm uses one population of learners and one population of evaluators.

Following the recent approach in coevolution known as *Pareto-Coevolution* [10, 17], learners are evaluated by viewing their outcomes against the tests in the test population as *objectives* in the sense of Evolutionary Multi-Objective Optimization (EMOO) [9]. Evaluators aim to detect *differences* between learners so as to allow for informed decisions in learner replacement. An evaluator is said to *distinguish* between two learners if it assigns a higher outcome to one learner than to the other. The notion of distinctions was introduced by Sevan Ficici in [11].

We will use a symmetric test problem with three possible outcomes: win (1), draw (0), or lose (-1). In this problem, the outcome of an interaction $G(a, e)$ between a learner a and an evaluator e returns 1 if and only if the outcome is a win for the learner. Since we are using the experiments as a model of practical problems, it should be sufficiently difficult for evaluators to make distinctions. Therefore, we will only say that an evaluator distinguishes between two learners in the case of win/lose outcomes. Thus, whether an evaluator e makes a distinction between learners a and b is defined as follows:

$$dist(e, a, b) \quad \Longleftrightarrow \quad G(a, e) = 1 \ \wedge \ G(b, e) \neq 1 \tag{1}$$

Evaluators are evaluated by using all potential distinctions between two learners in the learner population as objectives. Since a learner cannot be distinguished from itself, this results in a set of $n_l^2 - n_l$ objectives for a population of n_l learners. In addition, the n_l objectives of obtaining a high score against a learner are used, resulting in a total of n_l^2 objectives. Evaluation is based on the above choice of objectives for learners and evaluators, and employs the *Pareto-dominance* relation, see e.g. [9].

The DELPHI algorithm operates as follows. The learner and evaluator populations are initialized using random individuals. Next, the following cycle is repeated until a target performance criterion is met. A generation of learners is generated using the available operators of variation, typically mutation and crossover. For each new learner, we consider in turn whether it dominates an existing learner; if so it replaces it, if not it is discarded. Next, the same procedure is applied to the evaluator generation, except that a newly generated evaluator can only replace its *parent* if it dominates it; this additional requirement promotes diversity, and is based on Mahfoud's *Deterministic Crowding* procedure [14].

3 Exploration versus Reliability: an Insurmountable Tradeoff?

The strict replacement criterion of Pareto-dominance allows the DELPHI algorithm to achieve reliable progress in difficult test problems, such as the five-dimensional

COMPARE-ON-ONE problem with mutation bias [7]. However, the DELPHI algorithm can only make progress if, given a current population, a better individual can be generated within a single step of mutation or crossover. If the fitness landscape of a problem contains local optima given the available operators, identifying a better individual may require *multiple* steps of mutation or crossover that do not appear beneficial. To study how coevolutionary methods can be developed that combine reliability with exploration, we define a test problem that *requires* exploration in order to be solved.

The test problem is based on the COMPARE-SYMMETRIC function, which is defined as follows:

$$G_{\text{sym}}(a,e) = \begin{cases} 1 & \text{if} \quad \forall i : a_i \geq e_i \quad \wedge \quad \not\forall i : a_i = e_i \\ -1 & \text{if} \quad \forall i : e_i \geq a_i \quad \wedge \quad \not\forall i : a_i = e_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where a is a learner, e is an evaluator, both learners and evaluators are real-valued vectors, and x_i denotes the value of individual x in dimension i. In its basic form, this problem does not require exploration, assuming for instance the standard mutation operator as the operator of variation. We can introduce a need for exploration by *discretizing* the values of each individual before the outcome of the interaction function is determined; this results in plateaus in genotype-space on which all individuals have equal outcomes, so that progress can only be detected when the next plateau is reached. If the discretization interval is larger than the mutation range, then from certain regions in the space *multiple* steps will be required to arrive at an improved location, as desired. Discretization is performed by applying the following function to each value of the individual: $d(x) = \delta \lfloor \frac{x}{\delta} \rfloor$.

To test the effectiveness of this procedure in making the COMPARE-SYMMETRIC difficult for methods with limited exploration, we study the performance of the DELPHI algorithm on both the normal and the discretized version of the COMPARE-SYMMETRIC problem. Individuals are initialized by choosing each value randomly from $[0, 0.125]$. New individuals are generated using two-point crossover (50%) or mutation (50%). Mutation randomly chooses a dimension and adds a constant chosen uniformly from [-0.05,0.2], and is applied twice when used; this ensures that an increase in one dimension is often accompanied by a decrease in another dimension, which substantially complicates the problem of detecting progress based on interaction outcomes. Again, this choice is made to model difficulties faced by coevolutionary algorithms in practice. The experiments use the 3-dimensional COMPARE-SYMMETRIC problem. For the discretized version of this problem, $\delta = 0.25$. Both the learner and the evaluator population are of size 50. The curves show the best individual in the population. Score are determined by the lowest dimension of an individual, and averaged over 50 runs.

Figure 1 (left) shows that while DELPHI performs well on the continuous version of COMPARE-SYMMETRIC, it fails entirely when the problem is discretized. This confirms that the discretization has the desired effect of making the problem unsolvable for methods that only performs a single step look-ahead. In the following, we will investigate whether methods can be developed that can address the discretized version of the problem. To this end, we consider how the aims of

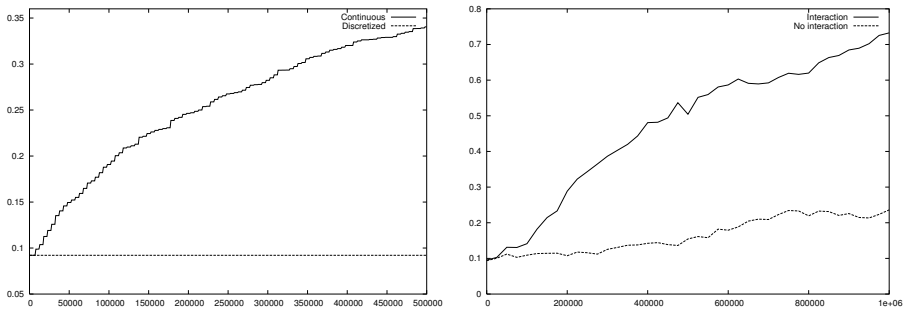


Figure 1: Left: Performance of Delphi on the compare-symmetric problem. Since exploration is required for the discretized problem, Delphi is unable to progress. Right: Performance of the setups with multiple subpopulations with varying degrees of exploration. While the non-interacting subpopulations already achieve some progress on the discretized problem, the interacting version greatly improves over this performance.

reliability and exploration may be combined in a single method.

4 Combining Exploration and Reliability

We face the problem of deciding whether to accept or reject a new individual, based on the objective values of the two individuals. A reliable choice that avoids regress is to accept the new individual only if it dominates the current individual. The other extreme of a maximally explorative choice is to always accept the replacement. Selection strategies in between these two extremes can be characterized by the degree to which they perform exploration.

Using the Boltzmann distribution, we can define a selection strategy with a *tunable* degree of exploration, so that selection methods from anywhere in the spectrum of exploration can be automatically produced. The degree of exploration is regulated by a *temperature* parameter T. Specifically, an individual with value v will be accepted into the population with probability $\frac{e^{\frac{v}{T}}}{e^{\frac{v_{max}}{T}}}$.

To distinguish between dominating, non-dominated, and dominated individuals, these classes are assigned values of 100, 40-60, and 10 respectively. The value of a non-dominated individual is determined by the fraction of objectives for which has a high value.

We will investigate a setup consisting of five matched pairs of learner and evaluator subpopulations, where each learner subpopulation is evaluated as before on its corresponding evaluator subpopulation, vice versa. Subpopulation pairs with a varying degree of exploration can now easily be generated by choosing a different temperature for each subpopulation pair.

The use of a range of different temperatures makes it likely that some temperature will provide the 'right' degree of exploration. However, there is no guarantee

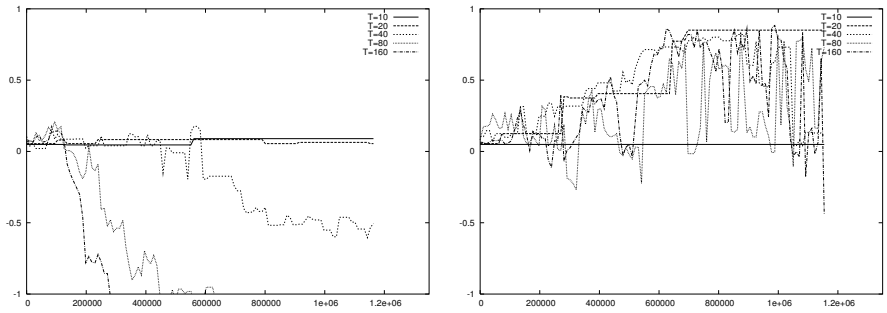


Figure 2: Performance per subpopulation (first run). As the difference between the graphs shows, the use of interaction greatly enhances performance; explorative subpopulations are able to generate improvements, and can avoid regress by using the stricter subpopulations as an archive. The stricter subpopulations provide reliability, but are also able to progress by importing improvements from explorative subpopulations. In this manner, the desirable aims of combination of exploration and reliability are combined.

that any single degree of exploration addresses the problem of balancing exploration and reliability. The central idea of this paper is therefore to let the subpopulations *interact*; if explorative subpopulations can identify improvements and these improved individuals can migrate to more reliable, less explorative subpopulations, the two aims of exploration and reliability may be combined. To achieve this, we let each subpopulation consider all other subpopulations to choose the parents used to produce offspring.

Figure 1 (right) shows the results of using non-interacting subpopulations. The curves show the best minimum value of an individual over all subpopulations. While DELPHI was seen to stall for this problem due to its lack of exploration, the use of varying degrees of exploration already results in some progress. However, as noted, the use of independent subpopulations does not guarantee that exploration and stability will be combined. The second curve in Figure 1 presents results with the proposed scheme for combining exploration and reliability. As the graph demonstrates, this method greatly improves performance, and achieves substantial progress on the difficult discretized COMPARE-SYMMETRIC problem.

To see whether the proposed method can indeed combine exploration and reliability, we also plotted the performance of each subpopulation separately, see Figure 2. This figure clearly shows the operation of the proposed method, and clarifies *how* it can combine exploration and stability. In the non-interacting setup, highly explorative subpopulations can achieve arbitrarily bad performance by accepting too many detrimental replacements. The strict, less explorative subpopulations for this method are not able to improve very much, as they are unable to generate new, improved individuals. In the setup *with* interaction, the subpopulations behave very differently, even though they have exactly the same temperature values as in the first setup. The strict subpopulations are able to progress every

once in a while, apparently through the use of genetic material from other subpopulations, as this is the only difference with the control setup. They thereby function as a form of *archives* that accept improvements when they are made, but do not produce improvements themselves. Interestingly, the behavior of the highly explorative subpopulations also changes; whereas formerly these were likely to regress, they can now use material from the more strict subpopulations to undo such regress. In summary, the interaction between subpopulations with different degrees of exploration makes it possible to obtain the desirable combination of exploration and reliability in coevolutionary algorithms.

5 Conclusions

Recently, theoretical advances in coevolution have suggested criteria for the evaluation of the tests used to evaluate learners. A first algorithm based on these ideas, named DELPHI, is able to achieve progress on all underlying objectives of challenging test problems, but is limited in that it does not perform exploration. We have investigated how exploration and reliability may be combined. It was found that a setup with interacting subpopulations with varying degrees of exploration is able to achieve this desirable goal. We hope this result may bring us closer to the goal of theoretically informed, practical algorithms for coevolution.

References

[1] Robert Axelrod. The evolution of strategies in the iterated prisoner's dilemma. In Lawrence Davis, editor, *Genetic Algorithms and Simulated Annealing*, Research Notes in Artificial Intelligence, pages 32–41, London, 1987. Pitman Publishing.

[2] Nils Aall Barricelli. Numerical testing of evolution theories. Part I: Theoretical introduction and basic tests. *Acta Biotheoretica*, 16(1–2):69–98, 1962.

[3] Anthony Bucci and Jordan B. Pollack. Order-theoretic analysis of coevolution problems: Coevolutionary statics. In *Proceedings of the GECCO-02 Workshop on Coevolution: Understanding Coevolution*, 2002.

[4] Anthony Bucci and Jordan B. Pollack. A mathematical framework for the study of coevolution. In *Foundations of Genetic Algorithms (FOGA-2002)*, (To appear).

[5] D. Cliff and G. F. Miller. Tracking the Red Queen: Measurements of adaptive progress in co-evolutionary simulations. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Proceedings of the Third European Conference on Artificial Life: Advances in Artificial Life*, volume 929 of *LNAI*, pages 200–218, Berlin, 1995. Springer.

[6] Edwin D. De Jong and Jordan B. Pollack. Principled Evaluation in Coevolution. Technical Report CS-02-225, Brandeis University, May 31, 2002.

[7] Edwin D. De Jong and Jordan B. Pollack. Learning the ideal evaluation function. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-03*, 2003.

[8] Edwin D. De Jong and Jordan B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, (To appear).

[9] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley & Sons, New York, NY, 2001.

[10] Sevan G. Ficici and Jordan B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Julian Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.

[11] Sevan G. Ficici and Jordan B. Pollack. Pareto optimality in coevolutionary learning. In Jozef Kelemen, editor, *Sixth European Conference on Artificial Life*, Berlin, 2001. Springer.

[12] Charles J. Geyer and Elizabeth A. Thompson. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90(431):909–920, 1995.

[13] D. W. Hillis. Co-evolving parasites improve simulated evolution in an optimization procedure. *Physica D*, 42:228–234, 1990.

[14] Samir W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, May 1995. IlliGAL Report 95001.

[15] Jordan B. Pollack and Alan D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(1):225–240, 1998.

[16] Karl Sims. Evolving 3D morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 28–39, Cambridge, MA, 1994. The MIT Press.

[17] Richard A. Watson and Jordan B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Julian Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.

[18] Richard A. Watson and Jordan B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-01*, pages 702–709, San Francisco, CA, 2001. Morgan Kaufmann.

On Probabilistic Connections of Fuzzy Systems

Uzay Kaymak Jan van den Berg

Erasmus University Rotterdam, Faculty of Economics
P. O. Box 1738, 3000 DR, Rotterdam, the Netherlands
e-mail: {kaymak, jvandenberg}@few.eur.nl

Abstract

We consider function approximation by fuzzy systems. Researchers outside the fuzzy systems community have many times criticized function approximation by fuzzy systems as being heuristics driven and unrelated to the probabilistic nature of the uncertainty. We demonstrate that the probabilistic nature of uncertainty is taken into account by probabilistic fuzzy systems. Furthermore, these systems take also fuzzy uncertainty into account by their fuzzy partitioning of input and output spaces. We discuss how additive reasoning in probabilistic fuzzy systems leads to the estimation of conditional probability densities, and how the fuzzy systems compute the expected value of this conditional density function. We show that some of the most commonly used fuzzy systems can compute the same expected output value and we derive how their parameters should be selected in order to achieve this goal. Hence, we provide reasons for the success of fuzzy systems in function approximation.

1 Introduction

Approximation of unknown functions from sampled data is an important activity in modern modelling and systems theory. With the advent of modern computer systems, the costs of data collection and storage have been reduced significantly. However, it has become equally important to develop models from the data, which have sufficient generalization power and can describe the underlying process with accuracy despite the nonlinearity and the complexity of these processes. The machine learning community has responded to this need by developing various methods such as neural networks [1], support vector machines [2] and fuzzy systems [6], which can be used for nonlinear function approximation.

Amongst the systems that have universal approximation capability, fuzzy systems have attracted particular interest due to their ability to provide linguistic descriptions of the modelled process. Encouraged by their success in practical applications, fuzzy sets community have proposed various rule base structures and reasoning mechanisms for fuzzy systems (e.g. [9, 10]), putting the emphasis on the modelling of the linguistic uncertainty and the interpolation capability of fuzzy systems. Many researchers outside the fuzzy set community, however, have felt uneasy about the success of fuzzy systems for function approximation, partly because the connection of these systems to the probabilistic nature of uncertainty in many data sets was unclear (see e.g. the panel discussion by the representatives of three European Networks of Excellence on fields related to computational intelligence in [3]). Fuzzy systems are thus often criticized as being heuristic systems without clear connections to probability theory.

Since fuzzy systems are known to be universal approximators [7], it is reasonable to assume that they lend themselves for probabilistic analysis, just like other universal approximators known from the literature. The question that needs to be answered is whether fuzzy systems are able to estimate conditional probability density functions (pdf's), and in particular, whether they are able to estimate the conditional expected output values for a given system. If the answer is positive, this can explain the success of fuzzy systems for function approximation.

Various researchers have studied the relation between probabilistic and fuzzy systems. Dubois and Prade have studied the relation between the possibility theory and the probability theory [4]. However, fuzzy systems for function approximation are not based on the possibilistic interpretation of the fuzzy sets. Kosko has analyzed the relation of fuzzy systems to probabilistic systems [8]. He finds a connection between fuzzy systems and probabilistic systems, but his argument is mainly based on the mathematical similarity of center-of-gravity defuzzification [6] to the computation of the expectation in probability theory. However, since the concept of membership is different than the concept of probability density, his results are not conclusive. In this paper, we follow an approach similar to [16]. We consider the relation of fuzzy systems for function approximation to the probabilistic uncertainty in the data within a framework of probabilistic fuzzy systems, which deal explicitly with two types of uncertainty (fuzziness or linguistic uncertainty and probabilistic uncertainty) based on probability measures for fuzzy events. We show that probabilistic fuzzy systems estimate conditional pdf's for the output variable, given the inputs to the system. We provide an additive reasoning mechanism for this purpose. We derive simple expressions for computing the expected output of a probabilistic fuzzy system. We show that zero-order Takagi–Sugeno (TS) deterministic fuzzy systems use the same expressions for reasoning. Hence, its parameters can be selected such that its output is equal to the conditional expected value of the identified function.

The outline of the paper is as follows. In Section 2, we give an overview of the concept of probability of fuzzy events, which is at the basis of probabilistic fuzzy systems. We introduce probabilistic fuzzy systems in Section 3 and we discuss how reasoning can be made with these systems. An additive reasoning mechanism is introduced. It is explained how conditional expected output of the system can be computed. In Section 4, the relation of probabilistic fuzzy systems to deterministic fuzzy systems is considered. It is shown that the output of both systems can be equivalent in certain cases. We discuss in Section 5 several issues related to our findings, and conclude the paper in Section 6.

2 Probability of fuzzy events

Probabilistic fuzzy systems are based on the concept of the probability of a fuzzy event, as defined by Zadeh [17]. In the following, we give a brief introduction to the theory of probability measures of fuzzy events and we cite several results that we will need in the following sections. The material in this section assumes a random scalar variable x defined on a continuous sample space X. The results for discrete variables and vector variables are analogous.

A compact subset Γ of X defines an event, and its probability $\Pr(\Gamma)$ is found by

integrating the probability density function $f(x)$ as

$$\Pr(\Gamma) = \int_{x\in\Gamma} f(x)dx = \int_{-\infty}^{\infty} \chi_\Gamma(x) f(x)dx\,, \tag{1}$$

where $\chi_\Gamma(x)$ is the binary characteristic function for the event Γ such that $\chi_\Gamma(x) = 1 \Leftrightarrow x \in \Gamma$ and $\chi_\Gamma(x) = 0$ otherwise. In other words, the probability of an event is the expectation of its characteristic function.

By replacing the characteristic function in (1) with a membership function $\mu(x)$: $X \rightarrow [0, 1]$, the probability measure for crisp events can be extended to a probability measure for fuzzy events. In this case, the probability of a fuzzy event A is found by taking the expectation of the membership function as

$$\Pr(A) = \int_{-\infty}^{\infty} \mu_A(x) f(x)dx = \mathrm{E}(\mu_A(x))\,. \tag{2}$$

This result allows us to assess the probability of a fuzzy event from sampled data by using standard expectation estimators such as the arithmetic mean [15], provided that the samples x_p, $p = 1, \ldots, P$, come from a well-defined sample space X, i.e. the fuzzy sets on X form a proper fuzzy partition [15].

In Section 3, we will need estimates for the conditional probability for a fuzzy event A, given another fuzzy event B. This can be found by [15]

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)} \approx \frac{\sum_{p=1}^{P} \mu_A(x_p)\mu_B(x_p)}{\sum_{p=1}^{P} \mu_B(x_p)}\,, \tag{3}$$

where the intersection of two fuzzy events is modelled by the product t-norm.

In classical probability theory, we can approximate a probability density function with a finite support by scaling the characteristic functions of crisp events for a disjoint cover of the support. Such an approximation is called a histogram. Assuming we partition the support into disjoint sets Γ_j, $j = 1, \ldots, J$, the probability density function $f(x)$ is approximated by

$$f(x) \approx \hat{f}(x) = \sum_{j=1}^{J} \frac{\Pr(\Gamma_j)\chi_{\Gamma_j}(x)}{\int_{-\infty}^{\infty} \chi_{\Gamma_j}(x)dx}\,. \tag{4}$$

Similarly, one can approximate the probability density function by scaling the membership functions of fuzzy events that form a proper fuzzy partition of the support as [15]

$$f(x) \approx \hat{f}(x) = \sum_{j=1}^{J} \frac{\Pr(A_j)\mu_{A_j}(x)}{\int_{-\infty}^{\infty} \mu_{A_j}(x)dx}\,. \tag{5}$$

Theorem 2.1 *Let X be a well-defined sample space partitioned into J fuzzy sets A_j, $j = 1, \ldots, J$. Then, $\int_{-\infty}^{\infty} \hat{f}(x)dx = 1$.*

Proof: Note that for a well-defined sample space, $\sum_{j=1}^{J} \Pr(A_j) = 1$. Then,

$$\int_{-\infty}^{\infty} \hat{f}(x) = \int_{-\infty}^{\infty} \sum_{j=1}^{J} \frac{\Pr(A_j)\mu_{A_j}(x)}{\int_{-\infty}^{\infty} \mu_{A_j}(x)dx} dx = \sum_{j=1}^{J} \Pr(A_j) \frac{\int_{-\infty}^{\infty} \mu_{A_j}(x)dx}{\int_{-\infty}^{\infty} \mu_{A_j}(x)dx} = 1. \tag{6}$$

Because of overlapping membership functions, fuzzy histograms appear to have nice approximation capabilities, better than crisp ones. We show this in figure Fig. 1 where the probability density function (pdf) of the standard normal distribution is approximated by a classical and by a fuzzy histogram using in both cases a partitioning in seven classes. For more details we refer to [14].

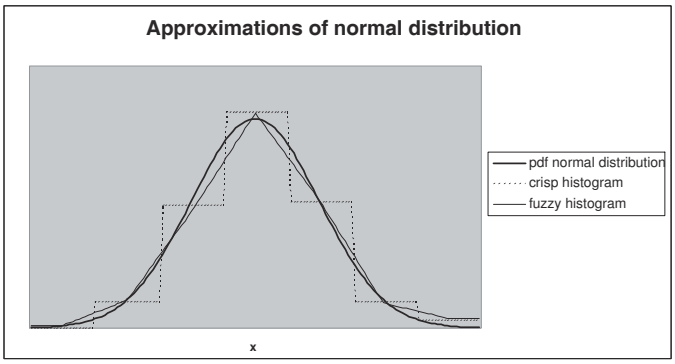


Figure 1: A fuzzy histogram better approximates a pdf than a crisp histogram.

3 Probabilistic fuzzy systems

Probabilistic fuzzy systems combine two different types of uncertainty, namely fuzziness or linguistic vagueness, and probabilistic uncertainty. In previous works, we have presented various types of probabilistic fuzzy systems with the corresponding reasoning schemes [5, 11, 12]. In this section, we present a more general formulation where the consequent of each rule is a conditional pdf, given the fuzzy antecedent of the rule. We assume that the function that is being approximated is a mapping $\mathbb{R}^n \rightarrow \mathbb{R}$ and that the rule consequents are defined on finite domain Y. This is not a serious restriction, since most function approximation takes place in a compact space.

A probabilistic fuzzy system consists of the rules R_q, $q = 1, \ldots, Q$, of the type

$$R_q : \text{If } \mathbf{x} \text{ is } A_q \text{ then } f(y) \text{ is } f(y|A_q)\,, \tag{7}$$

where $\mathbf{x} \in \mathbb{R}^n$ is an input vector, $A_q : X \longrightarrow [0,1]$ is a fuzzy set defined on X and $f(y|A_q)$ is the conditional pdf of the output given the fuzzy event A_q. If A_q had been crisp events, then only one of the rules would fire and hence only one of the conditional pdf's would be used. The system output can then be written as $\sum_{q=1}^{Q} \chi_q(\mathbf{x})\, f(y|A_q)$. In case of fuzzy events, multiple rules may fire and it is more appropriate to take a convex combination of rule outputs. We thus propose an additive reasoning mechanism that determines the output of fuzzy system as

$$f(y|\mathbf{x}) = \frac{\sum_{q=1}^{Q} \mu_{A_q}(\mathbf{x}) f(y|A_q)}{\sum_{q=1}^{Q} \mu_{A_q}(\mathbf{x})} = \sum_{q=1}^{Q} \beta_q(\mathbf{x}) f(y|A_q)\,, \tag{8}$$

where $\beta_q(\mathbf{x}) = \mu_{A_q}(\mathbf{x}) / \sum_{q=1}^{Q} \mu_{A_q}(\mathbf{x})$ represents the normalized degree of fulfillment of rule R_q. Note that this reasoning assumes that individual rules are connected with the "else" directive. The following theorem shows that the reasoning (8) returns a proper pdf.

Theorem 3.1 *Let $R = \cup_{q=1}^{Q} R_q$ be a fuzzy rule base consisting of the rules of type (7). Then, the reasoning scheme (8) computes a pdf, i.e. $\int_Y f(y|\mathbf{x})dy = 1$.*

Proof:

$$\int_Y f(y|\mathbf{x})dy = \int_{-\infty}^{\infty} \frac{\sum_{q=1}^{Q} \mu_{A_q}(\mathbf{x}) f(y|A_q)}{\sum_{q=1}^{Q} \mu_{A_q}(\mathbf{x})} dy = \frac{\sum_{q=1}^{Q} \mu_{A_q}(\mathbf{x}) \int_{-\infty}^{\infty} f(y|A_q)dy}{\sum_{q=1}^{Q} \mu_{A_q}(\mathbf{x})} = 1.$$

Therefore, if we know the pdf for each rule output, we can calculate the conditional pdf for each input vector $\mathbf{x}$. The expected conditional output of the probabilistic fuzzy system is given by the following theorem.

Theorem 3.2 *The expected conditional output of the probabilistic fuzzy system is given by the weighted average of the expected output of each rule, i.e.*

$$\mathrm{E}(y|\mathbf{x}) = \sum_{q=1}^{Q} \beta_q(\mathbf{x}) \mathrm{E}(y|A_q). \tag{9}$$

Proof:

$$\int_{-\infty}^{\infty} y \left[\sum_{q=1}^{Q} \beta_q(\mathbf{x}) f(y|A_q) \right] dy = \sum_{q=1}^{Q} \beta_q(\mathbf{x}) \int_{-\infty}^{\infty} y f(y|A_q)dy = \sum_{q=1}^{Q} \beta_q(\mathbf{x}) \mathrm{E}(y|A_q).$$

In general, the pdf's in the rule consequents are not available, and they must be estimated from the data. For this, we use the approximation in (5). Let J fuzzy classes C_j form a fuzzy partition of the compact output space Y. We now decompose each rule to provide a stochastic mapping between its fuzzy antecedents and its fuzzy consequents. The rules are interpreted to have the following form.

$$\begin{aligned} \text{Rule } R_q\text{: If } \mathbf{x} \text{ is } A_q \text{ then} \quad & \underline{y} \text{ is } C_1 \text{ with } \Pr(C_1|A_q) \text{ and } \ldots \text{ and} \\ & \underline{y} \text{ is } C_J \text{ with } \Pr(C_J|A_q). \end{aligned} \tag{10}$$

Now, by using the additional parameters $\Pr(C_j|A_q)$ and the characterization (5), we can compute the expected conditional output of the fuzzy system by using the following theorem.

Theorem 3.3 *Let R_q be the rules of a probabilistic fuzzy system. Let the output domain Y of the probabilistic fuzzy system be partitioned into J fuzzy classes C_j such that Y forms a well-defined sample space. Then the expected conditional output of the fuzzy system is approximated by*

$$\mathrm{E}(y|\mathbf{x}) \approx \sum_{q=1}^{Q} \sum_{j=1}^{J} \beta_q(\mathbf{x}) \Pr(C_j|A_q) z_j \,, \tag{11}$$

where z_j is the centroid of the jth output fuzzy set and $\Pr(C_j|A_q)$ is computed according to (3).

Proof: For the proof of this theorem, we compute $\int_Y y f(y|\mathbf{x})dy$, and we first substitute (8) for $f(y|\mathbf{x})$ and then (5) for $f(y|A_q)$.

$$\begin{aligned} \mathrm{E}(y|\mathbf{x}) &\approx \sum_{q=1}^{Q} \beta_q(\mathbf{x}) \int_Y y \sum_{j=1}^{J} \Pr(C_j|A_q) \frac{\mu_{C_j}(y)}{\int_Y \mu_{C_j}(y)dy} dy \\ &= \sum_{q=1}^{Q} \beta_q(\mathbf{x}) \sum_{j=1}^{J} \Pr(C_j|A_q) \frac{\int_Y y\mu_{C_j}(y)dy}{\int_Y \mu_{C_j}(y)dy} = \sum_{q=1}^{Q} \sum_{j=1}^{J} \beta_q(\mathbf{X}) \Pr(C_j|A_q) z_j \,, \end{aligned}$$

where z_j is the centroid of the fuzzy set C_j defined by

$$z_j = \frac{\int_Y y\mu_{C_j}(y)dy}{\int_Y \mu_{C_j}(y)dy} \,. \tag{12}$$

For modelling purposes, the parameters $\Pr(C_j|A_q)$ and z_j can be computed once offline. The evaluation of the expected output then requires the evaluation of $\beta_q(\mathbf{x})$ for a given $\mathbf{x}$ and the evaluation of (11), which can be very fast.

4 Relation to deterministic fuzzy systems

In this section, we consider the relation of the probabilistic fuzzy system described in Section 3 to deterministic fuzzy systems. In particular, we are interested in the relation between the expected output of a probabilistic fuzzy system and the deterministic output of a zero-order Takagi–Sugeno system [10].

Theorem 4.1 *A zero-order Takagi–Sugeno fuzzy system with Q rules with antecedent fuzzy sets A_q and consequent parameters c_q computes the expected value of the conditional pdf provided that the parameters c_q are equal to the expected defuzzified output of the probabilistic fuzzy system, i.e. provided that $c_q = \sum_{j=1}^{J} \Pr(C_j|A_q) z_j$.*

Proof: The proof is provided by re-arranging (11) and comparing it to the output of a zero-order Takagi–Sugeno system. The output of a zero-order deterministic Takagi–Sugeno system is given by

$$y* = \sum_{q=1}^{Q} \beta_q(\mathbf{x}) c_q \,. \tag{13}$$

Re-arranging (11) gives

$$\mathrm{E}(y|\mathbf{x}) = \sum_{q=1}^{Q} \beta_q(\mathbf{x}) \sum_{j=1}^{J} \Pr(C_j|A_q) z_j = \sum_{q=1}^{Q} \beta_q(\mathbf{x}) c_q \,,$$

with

$$c_q = \sum_{j=1}^{J} \Pr(C_j|A_q) z_j. \tag{14}$$

Therefore, by selecting the consequent parameters of the TS model in a specific way, one can approximate the expected output of the underlying system that has generated the data. Note that in many cases the parameters of TS fuzzy systems are optimized to minimize an error function, and hence optimality can be achieved in practical situations. This can explain the success of TS fuzzy systems for function approximation.

5 Discussion

The previous sections have shown that probabilistic fuzzy systems are able to approximate the conditional output pdf's for function approximation. Furthermore, the expected output of these systems is equal to the output of deterministic zero-order TS fuzzy systems, provided that the consequent parameters are selected according to (14). This property provides motivation for the success of additive fuzzy systems for function approximation. Note that in addition to the probabilistic nature of the data, probabilistic fuzzy systems let the analyst explicitly model linguistic concepts through the use of antecedent fuzzy sets A_q and the consequent fuzzy sets C_j. This allows the model to estimate the underlying probabilistic structure, while the model is calibrated to the linguistic description of the user. In addition to regular pdf's and conditional pdf's, probabilistic fuzzy models allow one to answer questions such as "what is the probability that the output is large given that the input is small" ($\Pr(C_j|A_q)$) or "what is the probability that the output is medium given a particular input" ($\Pr(C_j|\mathbf{x})$). Analyzing answers to these questions can provide additional information in a particular problem (see e.g. [13]).

Although we have discussed that the probabilistic fuzzy systems can approximate conditional pdf's, we have not analyzed the accuracy of this approximation. In general, the accuracy of the approximation of the conditional pdf's can be increased by increasing the number of consequent fuzzy sets C_j on the output domain. It is known that using a fuzzy partition already improves the approximation of the conditional pdf significantly [13]. Similarly, increasing the number of rules will improve the accuracy of interpolation between the rules.

6 Conclusions

Probabilistic fuzzy systems are able to approximate conditional pdf's, while at the same time calibrating the model to the linguistic conceptualization of the model maker. As such, they deal explicitly with both the fuzziness in the linguistic descriptions and the probabilistic uncertainty. We have proposed an additive reasoning scheme for probabilistic fuzzy systems. The expected output of these systems is shown to be computable from the probability of a consequent fuzzy event given an antecedent fuzzy event, the centroid points of the consequent fuzzy sets and the degree of fulfillment of the fuzzy rules. A zero-order TS fuzzy system can produce the same output as the expected output of a probabilistic fuzzy system provided that its consequent parameters are selected as the conditional expectation of the defuzzified output membership functions. Our results provide insight why additive deterministic fuzzy systems such as TS systems have proven to be so successful for function approximation purposes. Future research will concentrate on further study of probabilistic fuzzy systems.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[2] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, June 1998.

[3] Soundbytes from CoIL 2000. *Synergy*, (3):10–11, autumn 2000. Newsletter of CoIL, the Computational Intelligence and Learning Cluster.

[4] D. Dubois and H. Prade. Quantitative possibility theory and its probabilistic connections. In P. Grzegorzewski, O. Hryniewicz, and M. A. Gil, editors, *Soft Methods in Probability, Statistics and Data Analysis*, Advances in Soft Computing, pages 3–26. Physica Verlag, Heidelberg, 2002.

[5] U. Kaymak, W.-M. van den Bergh, and J. van den Berg. A fuzzy additive reasoning scheme for probabilistic Mamdani fuzzy systems. In *Proceedings of the 2003 IEEE International Conference on Fuzzy Systems*, volume 1, pages 331–336, St. Louis, USA, May 2003.

[6] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: theory and applications*. Prentice Hall, Upper Saddle River, 1995.

[7] B. Kosko. Fuzzy systems as universal approximators. *IEEE Transactions on Computers*, 43:1329–1333, 1994.

[8] B. Kosko. *Fuzzy Engineering*. Prentice Hall, Upper Saddle River, New Jersey, 1997.

[9] E. H. Mamdani and B. R. Gaines, editors. *Fuzzy Reasoning and its Applications*. Academic Press, London, 1981.

[10] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modelling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1):116–132, Jan./Feb. 1985.

[11] J. van den Berg, U. Kaymak, and W.-M. van den Bergh. Fuzzy classification using probability-based rule weighting. In *Proceedings of 2002 IEEE International Conference on Fuzzy Systems*, pages 991–996, Honolulu, Hawaii, May 2002.

[12] J. van den Berg, U. Kaymak, and W.-M. van den Bergh. Probabilistic reasoning in fuzzy rule-based systems. In P. Grzegorzewski, O. Hryniewicz, and M. A. Gil, editors, *Soft Methods in Probability, Statistics and Data Analysis*, Advances in Soft Computing, pages 189–196. Physica Verlag, Heidelberg, 2002.

[13] J. van den Berg, U. Kaymak, and W.-M. van den Bergh. Financial markets analysis by probabilistic fuzzy modelling. ERIM report series Research in Management, Erasmus Research Institute of Management, Erasmus University Rotterdam, the Netherlands, Apr. 2003. ERS-2003-036-LIS.

[14] J. van den Berg, U. Kaymak, and W.-M. van den Bergh. Financial markets analysis by using a probabilistic fuzzy modelling approach. *International Journal of Approximate Reasoning*, 2003. To appear.

[15] J. van den Berg, W. M. van den Bergh, and U. Kaymak. Probabilistic and statistical fuzzy set foundations of competitive exception learning. In *Proceedings of the Tenth IEEE International Conference on Fuzzy Systems*, volume 2, pages 1035–1038, Melbourne, Australia, Dec. 2001.

[16] G. C. van den Eijkel. *Fuzzy Probabilistic Learning and Reasoning*. Ph.D. thesis, Delft University of Technology, Delft University Press, Mekelweg 4, Delft, The Netherlands, Jan. 1999.

[17] L. A. Zadeh. Probability measures of fuzzy events. *J. Math. Anal. Appl.*, 23:421–427, 1968.

Inverting multi-layer perceptrons is easy

Wojtek Kowalczyk

Vrije Universiteit Amsterdam
Department of Computer Science
De Boelelaan 1081A, 1081HV Amsterdam

Abstract

We present an approximate algorithm for reconstructing internals of multi-layer perceptrons from membership queries. The key component of the algorithm is a procedure for reconstructing weights from linear threshold units. We prove that the approximation error, measured as the distance between the original and reconstructed weights, is dropping exponentially fast with the number of queries. The procedure is combined with a labelling strategy that involves solving multiple Linear Programming problems. This combination results in an algorithm that extracts internals of multi-layer perceptrons: the number of relevant hyperplanes, their parameters, and the logical structure. In practice, networks that compute combinations of 10-15 hyperplanes can be reconstructed in several minutes.

1 Introduction

In this paper we consider the following problem. Suppose that we are given a multi-layer perceptron that calculates a function f over the d-dimensional cube $[-1,1]^d$. (By perceptron we mean here a linear threshold unit that computes the characteristic function of a halfspace; multi-layer perceptrons compute boolean combinations of halfspaces.) Now the question is: how can we reconstruct the "internals" of such a black-box perceptron by asking a sequence of questions in the form: *what is the value of* $f(x)$ *on the given* x *from* $[-1,1]^d$*?* In particular, we are interested in finding the number of hyperplanes that correspond to the first layer, their parameters (weights), and their logical structure (which intersections of halfspaces are classified as *positive* and which as *negative*?).

The main result of this paper is an algorithm that automatically finds all these items. As we impose no restrictions on the precision of perceptron's weights, our algorithm returns an *approximate* model of the original perceptron.

We will start with proposing an algorithm for extracting weights from a single perceptron. This algorithm, called *orthogonal bracketing*, is shown to converge exponentially fast (in the number of queries) to true weights. It is used as the main component of a procedure that can handle arbitrary perceptrons. This procedure generates random queries to discover inconsistencies between the perceptron and its current approximation. Whenever an inconsistency is spotted, the bracketing procedure locates a new hyperplane that is responsible for it. Next, a number

of Linear Programming problems are generated and solved in order to find all non-empty intersections of detected halfspaces and their labels.

The actual implementation of the presented method can handle perceptrons with 10-15 nodes in the first hidden layer in a matter of minutes. The overall complexity of the algorithm is shown to be exponential in the number of such nodes, so reversing bigger networks might be practically intractable.

Our work is strongly related to research in the areas of computational learning theory and learning with queries. The problem of learning intersections of two homogeneous halfspaces was addressed by Baum, [4] who later generalized it to intersections of an arbitrary number of halfspaces, [3]. His results were further generalized and improved in a sequence of papers; Goldberg and Kwek, [6], present and extensive overview of these papers. They also tackle the problem of exact learning of convex polytopes under assumption of limited precision of weight representations. The literature on the subject of learning with queries is very extensive, [1], but it is mostly focused on learning in discrete spaces.

2 Extracting weights from a single perceptron

Let us consider a single perceptron with d inputs, i.e., a device that calculates a function $f : [-1, 1]^d \to \{-1, 1\}$ which is given by:

$$f(x) = sign(\sum_{i=1}^{d} w_i x_i + b).$$

Traditionally, the elements of vector $\mathbf{w} = (w_1, \ldots, w_d)$ are called *weights* and the parameter b is called the *bias*. Because the linear equation $\mathbf{w}^T x + b = 0$ determines a $(d-1)$-dimensional hyperplane H in R^d, the perceptron returns 1 on points that are above H and -1 on points that are below H. We will denote both halfspaces by H^+ and H^-, respectively. To make sure that the parameters $(\mathbf{w}, b)$ are uniquely determined by H we will assume that $\|\mathbf{w}\| = 1$. Then $\mathbf{w}$ is known to be normal to H and $|b|$ is the distance between the origin and H.

We are interested in an algorithm that asks a sequence of questions in the form: *what is the value of* $f(x)$ *on the given* x *from* $[-1, 1]^d$*?* and returns a vector $(\mathbf{v}, c)$ that closely approximates $(\mathbf{w}, b)$. Clearly, we would like our algorithm to ask as few questions as possible, while returning good approximations of true weights.

Clearly, to find parameters of a hyperplane H one only needs to find d linearly independent points that are on (or very close to) H. When such points are found, say, $p_1, \ldots, p_d$, all the weights can be calculated by solving the system of d linear equations $\mathbf{v}^T p_i + 1 = 0$, in $\mathbf{v} = (v_1, \ldots, v_d)$, and setting $c = 1/\|\mathbf{v}\|$ and $\mathbf{v} = \mathbf{v}/\|\mathbf{v}\|$. To find a point that is close to H it is enough to find a pair of points, p and q, that are on the opposite sides of H. We will call such a pair of points a *bracket*. Given a bracket (p, q) of size $\delta = \|p - q\|$ we can squeeze it exponentially fast by iterating the bisection operation:

```
r=(p+q)/2; if f(r)==f(p) then p=r; else q=r;
```

Clearly, after k iterations (what costs k queries) the resulting bracket has the size $\delta/2^k$. Thus, to find H, one could generate at random queries until d different brackets are formed; then the bisection operation should be applied to each bracket to make them smaller than $1/2^k$, where k is a program parameter. Finally, a hyperplane that passes through centers of all the brackets should be found. Unfortunately, such a "random bracketing" algorithm has several drawbacks. First, the d points that are used for finding the hyperplane might be positioned in such a way that the corresponding system of linear equations is ill-conditioned (they might be either too close to each other or almost linearly dependent). In such situations the approximation error might be relatively big. Second, when two brackets are relatively close to each other, i.e., when the ratio "bracket size":"distance" is big, there is a lot of "uncertainty" about the position of the true hyperplane, which may lead to big errors. Finally, it is very difficult to quantify the relation between the parameter k and the possible error, measured, for example, by $\|\mathbf{w} - \mathbf{v}\|$.

To avoid all these problems, the Orthogonal Bracketing Algorithm mimics the classical Gram-Schmidt orthogonalization process. Brackets are constructed sequentially in such a way that their centers form an orthonormal system.

Let us assume for a while that the hyperplane H is homogeneous, i.e., that the origin O belongs to H. Let $k > 0$ be an integer. We will construct $(d-1)$ points $p_1, p_2, \ldots, p_{d-1}$ in such a way that their distance to H is at most $1/2^k$, their distance to O is 1, and vectors $p_1, \ldots, p_{d-1}$ are orthogonal to each other. (Whenever there is no risk of confusion we slightly misuse the notation using the same symbols to denote points and vectors.) The construction of p_i's is inductive.

Initialization. Let p be an arbitrary vector of length 1. Due to symmetry (H passes through O) points p and $-p$ have different labels, i.e, $f(p) \neq f(-p)$. Let q be any vector of length 1 that is orthogonal to p. Clearly, $f(q) \neq f(p)$ or $f(q) \neq f(-p)$, so (p, q) or $(-p, q)$ is a bracket. Let us squeeze this bracket by iterating k times the spherical bisection operation:

```
r=(p+q)/2; r=r/||r||; if f(r)==f(p) then p=r; else q=r;
```

(here we assumed that $f(p) \neq f(q)$), and let p_1 denote its normalized center (after squeezing). Let us note that the distance between p_1 and H is smaller than $1/2^k$. Indeed, all the vectors that are produced by the squeezing procedure are on the unit circle that is located in a two dimensional space spanned by vectors p and q. The angular distance between p and q is $2\pi/4 < 2$ and it is divided by 2 with every bisection (k times), so the final bracket has the size smaller than $2/2^k$ and H must pass between its center (p_1) and one of its ends.

Inductive step. Let us suppose that we already have points $p_1, \ldots, p_i$, where $i < d-1$, that satisfy our constraints. To construct p_{i+1} let us consider an arbitrary vector p of length 1 that is orthogonal to vectors $p_1, \ldots, p_i$. Such p can be easily generated: take at random any vector x, set $p = x - p_1 < x, p_1 > - \ldots - p_i < x, p_i >$ and normalize it. (Since $i < d-1$ vector x is with probability 1 linearly independent from $p_1, \ldots, p_i$, so $\|p\| \neq 0$.) Let q be any vector of length 1 that is orthogonal to $p, p_1, \ldots, p_i$. By the same argument as used in the initialization step, one of the pairs: (p, q) or $(-p, q)$ must be a bracket. We squeeze it by k-

fold bisection and define p_{i+1} to be its normalized center. Moreover, by the same argument as before, the distance between p_{i+1} and H is smaller than $1/2^k$.

The fact that vectors $p_1, \ldots, p_{d-1}$ are orthonormal is crucial in the proof of the following theorem.

Theorem 1. *Suppose that H is given by $\mathbf{w}^T x = 0$, $\|\mathbf{w}\| = 1$. Let G denote a hyperplane $\mathbf{v}^T x = 0$, $\|\mathbf{v}\| = 1$, that passes through the points $O, p_1, \ldots, p_{d-1}$ that are determined by the above algorithm. Then we have:*

1. *For every $x \in G$ s.t. $\|x\| = 1$ there is $y \in H$ s.t. $\|x - y\| \leq \sqrt{d-1}/2^k$,*
2. *If $\sqrt{d-1}/2^k < \sqrt{3}/2$ then $\|\mathbf{w} - \mathbf{v}\| < \frac{2}{\sqrt{3}}\sqrt{d-1}/2^k$.*

Proof Let $x \in G$ be such that $\|x\| = 1$. Because vectors $p_1, \ldots, p_{d-1}$ form an orthonormal basis in G there exist $\alpha_1, \ldots, \alpha_{d-1}$ such that

$$x = \sum_{i=1}^{d-1} \alpha_i p_i \quad \text{and} \quad \sum_{i=1}^{d-1} \alpha_i^2 = 1.$$

Let $q_1, \ldots, q_{d-1} \in H$ be such that $\|p_i - q_i\| \leq 1/2^k$, for $i = 1, \ldots, d-1$, and let $y = \sum_{i=1}^{d-1} \alpha_i q_i$. Then, using Cauchy-Schwarz inequality, we have:

$$\|x - y\|^2 = (\sum_{i=1}^{d-1} \alpha_i (p_i - q_i))^2 \leq (\sum_{i=1}^{d-1} |\alpha_i| \|p_i - q_i\|)^2 \leq (\sum_{i=1}^{d-1} \alpha_i^2)(\sum_{i=1}^{d-1} \|p_i - q_i\|^2).$$

Thus $\|x - y\| \leq \sqrt{d-1}/2^k$.

To prove the second part let us consider a two-dimensional plane P that contains vectors v and w. Let $v' \in G \cap P$ and $w' \in H \cap P$ denote unit vectors that are orthogonal to v and w, respectively. Clearly, as v and w are normal to G and H, v' and w' do exit. Let h denote the height of the triangle $w'Ov'$ and let α denote the angle $v'Ow'$ and β the angle $Ov'w'$. From the first part of our theorem we have $h < \sqrt{d-1}/2^k$, and we assumed that $\sqrt{d-1}/2^k < \sqrt{3}/2$, therefore $h < \sqrt{3}/2$. Let us notice that for $h = \sqrt{3}/2$ the triangle $w'Ov'$ is equilateral, $\alpha = \pi/3$, and $\beta = \pi/6$. Moreover, when h is getting smaller, so does β. But $\|v' - w'\| = h/\cos\beta$ and $\sqrt{3}/2 < \cos\beta < 1$ for $0 < \beta < \pi/6$. Therefore $\|v - w\| = \|v' - w'\| < \frac{2}{\sqrt{3}}\sqrt{d-1}/2^k$.

The bounds derived above, especially the second one, are very useful: they provide information about the value of k under which the error is guaranteed to be smaller than a pre-specified value. As we can see, for a fixed k the error $\|v - w\|$ is bounded by $c\sqrt{d}$. However, the norm definition involves summation over d dimensions. Therefore the error bound "per dimension" (i.e., per weight) is limited by $c\sqrt{d}/d = c/\sqrt{d}$.

Let us return to the homogenity assumption that we have made at the beginning of this section. When H is not homogeneous we have to make two modifications. First, the initial bracket (or a point that plays the role of the origin) has to

be found. Here we apply the same trick as with the random bracketing algorithm: just make at most, say, 1000 random queries, until a bracket is found. If after 1000 queries no bracket is found we have a trivial case of a constant function. Otherwise, as soon as a bracket is found we squeeze it to a point (using a suitably large k) – let us call it p_0. Next, let us consider a ball with center in p_0 and radius r being the smallest coordinate of $|p_0|$, so the ball is fully included in $[-1,1]^d$. One can easily verify that the inductive construction of points $p_1, \ldots, p_d - 1$ is still valid: instead of O we use p_0 and instead of a unit ball we work on a ball with radius r. Moreover the bounds of Theorem 1 still hold.

3 Extracting internals from a MLP

Let us consider now a multi-layer perceptron with d inputs, i.e., a device that calculates a function $f : [-1,1]^d \to \{-1,1\}$. It is well-known that f can be represented as a boolean combination of halfspaces that are determined by linear threshold units from the first layer. Indeed, let these units define h hyperplanes $H_1, \ldots, H_h$, and let $\lambda \in \{-1,1\}^h$ be a sequence of labels that refer to two possible sides of each hyperplane. Then the value of f on the intersection $\mathbf{H}_\lambda = H_1^{\lambda_1} \cap \ldots \cap H_h^{\lambda_h}$ is constant; let us denote it by f_λ.

In other words, the function f is fully specified by h equations of the hyperplanes H_i

$$\mathbf{w}_i^T \mathbf{x} + b_i = 0, \quad \text{for} \quad i = 1, \ldots, h$$

and at most 2^h labels f_λ, for $\lambda \in \{-1,1\}^h$ (some $\mathbf{H}_\lambda$'s might be empty).

In the previous section we described the Orthogonal Bracketing algorithm which, after some modifications, can be used for finding H_i's (i.e., vectors $(\mathbf{w}_i, b_i)$). Now we will describe a labelling procedure that for given H_i's finds all labels f_λ.

3.1 Labelling procedure

Let us consider $\lambda \in \{-1,1\}^h$ and let $\mathbf{H}_\lambda = H_1^{\lambda_1} \cap \ldots \cap H_h^{\lambda_h}$. To find a label for $\mathbf{H}_\lambda$ we only need to find an arbitrary point $x \in \mathbf{H}_\lambda$ and apply f to it. This could be achieved by solving a system of h linear inequalities

$$\mathbf{w}_i^T \mathbf{x} + b_i > 0, \quad \text{or} \quad \mathbf{w}_i^T \mathbf{x} + b_i < 0$$

where choices are made depending on values of λ_i's. However, taking into account that our algorithm works with *approximations* of true H_i's, we would like to find x such that all the inequalities are satisfied by as large margin as possible–this should guarantee that x is in the "safe" region, far from boundaries. Fortunately, this extra requirement can be expressed as a Linear Programming problem that uses $h+1$ slack variables $z, z_1, \ldots, z_h$. Indeed, let us consider the following LP problem:

$$\text{Maximize } z \text{ subject to:}$$

$$\mathbf{w}_i^T \mathbf{x} + b_i - \lambda_i (z + z_i) = 0,$$

$$-1 \le x_i \le 1,$$

$$0 \leq z_i,$$
$$0 \leq z,$$
$$\text{for} \quad i = 1, \ldots, h.$$

We can see that any solution x of this problem lies in $\mathbf{H}_\lambda$ and that the variable z which measures the margin is maximized.

It should be noticed that z doesn't really reflect the Euclidean distance between x and the boundaries of $\mathbf{H}_\lambda$. A direct approach to maximizing this distance would immediately lead to a quadratic optimization problem which is computationally much more expensive than solving a simple LP. This issue is addressed in more depth in [5]. In practice, however, the LP approach is sufficiently powerful and the low computational costs make it very attractive. For example, using a popular LP solver that implements the interior-point method, LIPSOL, [7], we were able to solve labelling problems for 10 hyperplanes (i.e., generated and solved 1024 instances of LPs) in about 1-2 minutes (for d ranging from 20 to 100) on an ordinary PC with 1GHz processor.

3.2 Main Algorithm

The two procedures described earlier, orthogonal bracketing and labelling procedure, are the key components of the algorithm for extracting internals from multi-layer perceptrons. The algorithm uses two parameters: k, that controls the accuracy, and a threshold t that defines the termination condition.

The algorithm starts with an empty collection of hyperplanes and a constant labelling function that maps all inputs to 1, or -1, depending on the result of the first query, e.g., $f(0)$. Then random queries are made (up to t) to find out any inconsistency between the actual f and the current collection of hyperplanes and the corresponding labelling function. If within t queries no inconsistency is discovered the algorithm terminates. As we argued before, a suitably big value of t, e.g., 1000, is sufficient to provide high confidence in the (approximate) correctness of the result. If an inconsistency is spotted the algorithm switches to the bracketing mode to locate a hyperplane that is responsible for it. After finding approximate weights for this new hyperplane, it is added to the current set of hyperplanes and new labelling is recomputed. The whole process of finding inconsistencies and fixing them is repeated until t consecutive random queries provide answers that are consistent with the current labelling function.

The original bracketing algorithm requires some modifications. First of all, let us notice that an inconsistency is detected when two points x, y are found, such that $f(x) \neq f(y)$, and both x and y belong to the same $\mathbf{H}_\lambda$. In such a situation the pair (x, y) is a bracket for a yet unknown hyperplane. The squeezing procedure can be used for finding a starting point p_0 on this hyperplane, with accuracy $< 1/2^k$. Then the distance r from p_0 to the boundary of $\mathbf{H}_\lambda$ can be found (it is the minimum of the distances to all hyperplanes). Thus the ball with radius r and center in p_0 is fully included in $\mathbf{H}_\lambda$ and the orthogonal bracketing procedure can be started.

Although the orthogonal bracketing procedure always finds a solution in case of a single hyperplane, the possibility of multiple hyperplanes that pass through the ball may complicate things. Basically, there are two problems that may arise. The first one is that the search for next bracket may fail: now it is no longer the case that for any q on the ball, q and $-q$ have different labels. The second problem is at first sight even more severe: the procedure may converge and produce points that are on different hyperplanes. In this case, i.e., when the generated points $p_1, \ldots, p_{d-1}$ are located on two or more different hyperplanes, the hyperplane that is determined by these points is irrelevant. Fortunately, the labelling procedure takes care that the new labelling is always consistent with f, so although the new collection of hyperplanes may contain a bogus one, it will have no consequences for the final result. There is, however, a simple method for detecting such pathologies. When p and q belong to the same hyperplane then a small ball with center in $(p+q)/2$ intersects this hyperplane. In turn, testing if a ball intersects a hyperplane is simple: a few random queries on points that are on the ball should produce some alternating labels, otherwise the ball is most likely disjoint with it.

Thus we have two types of problems: the first one is easy to detect (the procedure doesn't converge), the second one, less harmful, can be detected by a heuristic test. In both cases the remedy is simple: divide r by 2 and restart the procedure. By decreasing the size of the ball we decrease the probability of encountering problems.

3.3 Complexity analysis

The presented algorithm detects hyperplanes one after another. Each time a new hyperplane is located the labelling procedure is called. It solves 2^i LPs, where i is the current number of hyperplanes. Therefore, if in total h hyperplanes are detected then $2^{h+1} - 1$ LPs have to be solved. In theory the LP solver works in polynomial time in the number of constraints and the number of variables, but for relatively small values of h and d the execution time may be considered to be constant. The time required by the bracketing procedure is proportional to khd and is so small (when compared to the rest) that it can be neglected.

Every solvable instance of LP generates a query, thus the number of queries that are generated by the labelling procedure is also bounded by 2^{h+1}. Additionally, for each hyperplane at most t random queries are made, followed by kd queries that are generated by the bracketing procedure (when we ignore pathological cases). Thus the total number of queries is about $h(t + kd) + 2^{h+1}$.

Concluding, both the time complexity and the number of queries are dominated by the 2^h term. Therefore, the presented algorithm is applicable only to networks with at most 10-20 nodes in the first hidden layer.

4 Conclusions

We presented two efficient algorithms for extracting weights from multi-layer perceptrons: the orthogonal bracketing algorithm for single perceptrons and a combined bracketing-labelling procedure for general case. The orthogonal bracketing

procedure was proved to produce approximations with error dropping exponentially fast with the number of queries. Experiments with an actual implementation of this procedure demonstrated that when the number of allowed queries is 10 (20 or 30) times the number of inputs, then weights are recovered with accuracy of about 3 (6 or 9, resp.) decimals. The complexity of the general procedure was shown to be bounded exponentially in the number of units in the first hidden layer (or, equivalently, in the number of involved hyperplanes). We believe that this bound is tight.

There are several issues that could be investigated further. We will outline only some of them. First, it would be interesting to provide deeper theoretical analysis of the presented algorithms and place them in the context of Valiant's PAC-learning theory or Angluin's framework for learning with queries. Second, more insights in the real-life performance of the presented algorithms should be gained. The algorithms are already implemented and some systematic experiments that are aimed at the issues of accuracy, speed, reliability, etc. are started. The results will be reported elsewhere. Third, we believe that the orthogonal bracketing procedure can be used as a starting point for constructing heuristic strategies for efficient labelling of training examples. Finally, a natural question to ask is the case of "ordinary" feed-forward networks with smooth activation functions (logistic sigmoid). Is the reverse engineering of such networks difficult?

References

[1] D. Angluin. Queries revisited. In T. Z. N. Abe, R. Khardon, editor, *Algorithmic Learning Theory , 12th International Conference, ALT 2001, Washington, DC, USA, November 25-28, 2001*, pages 12–31. Springer Verlag, 2001.

[2] S. Argamon-Engelson and I. Dagan. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, 1999.

[3] E. Baum. On learning a union of half spaces. *Journal of Complexity*, 6(1):67–101, 1990.

[4] E. Baum. Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks*, 2(1):5–19, 1991.

[5] R. Bramley and B. Winnicka. Solving linear inequalities in a least squares sense. *SIAM Journal on Scientific Computing*, 17(1):275–286, 1996.

[6] P. Goldberg and S. Kwek. The precision of query points as a resource for learning convex polytopes with membership queries. In *Proc. 13th Annu. Conference on Comput. Learning Theory*, pages 225–235. Morgan Kaufmann, San Francisco, 2000.

[7] Y. Zhang. User's guide to LIPSOL, Rice University, 1998.

Towards Data Mining in Large and Fully Distributed Peer-to-Peer Overlay Networks

Wojtek Kowalczyk Márk Jelasity A. E. Eiben

Vrije Universiteit Amsterdam
Department of Computer Science
De Boelelaan 1081A, 1081HV Amsterdam

Abstract

The Internet, which is becoming a more and more dynamic, extremely heterogeneous network has recently became a platform for huge fully distributed *peer-to-peer overlay networks* containing millions of nodes typically for the purpose of information dissemination and file sharing. This paper targets the problem of analyzing data which are scattered over a such huge and dynamic set of nodes, where each node is storing possibly very little data but where the total amount of data is immense due to the large number of nodes. We present distributed algorithms for effectively calculating basic statistics of data using the recently introduced *newscast model of computation* and we demonstrate how to implement basic data mining algorithms based on these techniques. We will argue that the suggested techniques are efficient, robust and scalable and that they preserve the privacy of data.

1 Introduction

With the rapid increase in the number of computers connected to the Internet and the emergence of a range of mobile computational devices which might soon be equipped with mobile IP technology, the Internet is converging to a more dynamic, huge, extremely heterogeneous network which nevertheless provides basic services such as routing and name lookup. This platform is already being used to support huge, fully distributed *peer-to-peer overlay networks* containing millions of nodes typically for the purpose of information dissemination and file sharing [8]. Such fully distributed systems generate immense amounts of data. Analyzing this data can be interesting from both scientific and business purposes. Among other applications, this environment is a natural target for distributed data mining [10].

In this paper we would like to push the concept of distributed data mining to the extreme. The motivations behind distributed data mining include the optimal usage of available computational resources, privacy and dependability by eliminating critical points of service. We will adopt the harshest possible constraints on the distribution of data and the elements of the network and demonstrate techniques which can still provide useful information about the distributed data effectively and dependably.

There are two constraints that we will adopt. The first is that all nodes are allowed to hold as few as one single data instance. This can be viewed as an

extremum of horizontal data distribution. The second is another extremum: there is practically no limit on the number of nodes. The only requirement is that in principle each pair of nodes could communicate directly which holds if the nodes are on the Internet with a (not necessarily fixed) IP address.

Furthermore, we will concentrate on two other very important aspects. The first is data privacy, the second is the dynamic nature of the underlying network: nodes can leave the overlay network and new nodes can join it.

To achieve our goal we will work in the newscast model of computation [5]. This model is built on a lower layer, an epidemic protocol for disseminating information and group membership [4], and it provides a simple interface for applications. The advantage of the model is that due to the robustness and scalability of the epidemic protocol it is built on, the applications of the newscast model of computation inherit this robustness and scalability and can target the kinds of distributed networks described above.

2 The Newscast Model of Computation

The newscast model has been developed as part of the European FP5-IST DREAM project [9]. The newscast model of computation is implemented by a probabilistic epidemic protocol for information and membership dissemination. This protocol provides a dependable, scalable and robust way of maintaining a connected overlay network and disseminating information among its members effectively. Here we do not discuss this protocol since it is not necessary for understanding the paper. The interested reader should consult [5]. Information about related work can be found in [2, 6].

During the discussion of the newscast model of computation we will make further simplifications avoiding the technical details and focusing only on those properties that we apply when developing our algorithms.

The two main concepts of the model are the *collective of agents* and the *news agency*. Computation is performed by the agents that might have their own data storage, processor and I/O facilities. The agents communicate through the news agency according to a special schedule which is orchestrated by the news agency. It is very important to stress here that although the news agency plays the role of a server in the model, it is a purely *virtual* entity and the actual implementation of its functionality at the protocol level is a fully distributed peer-to-peer solution.

The communication schedule is organized into *cycles*. In each cycle the news agency collects exactly one *news item* from all the agents. At the same time it delivers to every agent a random sample of c news items that were collected in the previous cycle.

Even though we do not discuss the protocol here, note that since agents receive only the news content but no information about the sender, the system can stay completely anonym so privacy is not violated. The actual protocol that implements this model can effectively act as a "remailer", where the origin of a given item is hard to track down.

To shed some more light on how to develop applications for the model, we

present an easily comprehensible yet interesting example. Let us assume that the collective contains n agents, and each agent i knows a single number a_i. The task is to find the maximum of these numbers $a^* = \max_{i=1}^{n} a_i$. The following two-liner, which will be common to all agents, will solve this problem.

```
NewsItem newsUpdate(news[]) {
    myMax = max(myMax, a, news[1],...,news[c]);
    return myMax;}
```

where $a = a_i$ for agent i.

It is important to note that reading the output of the algorithm is possible for *all* agents, so there is no need for a specific user terminal or service to extract the output. Although there is no signal that informs the agents that the value is found, using the theory of epidemic algorithms [1] it can be proven that all agents will hear about the final solution very quickly. The trick is that from the point of view of a true maximum value the algorithm is in fact an effective broadcasting mechanism, since all agents will keep returning it after they have seen it at least once. So the maximum value spreads exactly like an epidemic, "infecting" a quickly growing number of agents. Let as assume that p_i is the probability that a given agent is not infected in cycle i. The probability that a given agent is not infected in cycle $i+1$ is given by $p_{i+1} = p_i p_i^c$ since it had to be uninfected in cycle i and none of its c samples in the news update must be infective. The initial value $p_0 = (1 - 1/n)$. It is clear that p_i decreases extremely fast.

3 Calculating Basic Statistics

Let us consider a system of n agents that form a newscast network, and let each agent store one number–its own value. Our objective is to program these agents in such a way, that they will collectively find, within very few cycles, the mean of all values (or a good approximation of it). In this section we will present three algorithms for this task: basic averaging, (BA), systematic averaging (SA), and cumulative averaging (CA). These algorithms, although based on the same idea, have different properties with respect to convergence speed, accuracy and adaptivity.

The ability of finding the mean is central for implementing some basic data mining algorithms within the newscast framework. In Section 4 we will demonstrate how the process of finding the mean can be adopted for finding other statistics, like conditional probabilities, information gain, Gini index, etc. – the key elements for building various classification procedures like Naive Bayes and decision trees.

To simplify the statistical analysis of the behavior of our algorithms we will assume that $c = 2$, i.e., that news that are distributed by the news agency always consist of 2 news items. It should be noticed that in practice the value of c is usually much bigger than 2 (e.g., in our experiments we used $c = 20$) which yields much faster convergence rates than our theoretical bounds.

3.1 Basic Averaging

Probably this is the simplest algorithm for finding the mean. During the first cycle (when no news are available) every agent publishes its own value. In this way the news agency gets a copy of all values to be averaged. Next, all agents switch to the "averaging mode": whenever they receive news they calculate the average of all news items and publish it. More formally, agent's behavior – the `newsUpdate(news[])` function (where `news[]` refers to the list of news items, each of them being a single number) – is defined as follows:

```
NewsItem newsUpdate(news[]) {
         if (news[] is empty) return own value;
         else return the average of elements in news[];}
```

The rationale behind the algorithm is based on the following observation: if we are given a set of numbers and replace two of them by their average then the overall mean will not change, but the variance will decrease. Therefore, in every cycle the news agency receives a collection of numbers that (on average) has the same mean as the mean of the original set, but the variance will be getting smaller and smaller. As a matter of fact, the variance is dropping exponentially fast with the number of cycles: every cycle reduces the variance by factor 2. Indeed, in a single cycle n pairs of numbers are drawn at random (we assumed $c = 2$), and consequently each pair is averaged. This can be modeled by a random variable $(X + Y)/2$, where X and Y are independent random variables that take values in V – the set of values kept by the news agency – with each value having the same chance. Clearly, we have: $E[(X + Y)/2] = E[X] = E[Y] = E[V]$ and $Var((X + Y)/2) = Var(X)/4 + Var(Y)/4 = Var(V)/2$, where $E[.]$ denotes the expected value (mean) and $Var(.)$ the variance of a random variable (so we are misusing a bit the notation, as V is not a random variable).

As said earlier, the newscast model that we are working with is an idealization of the real model that works in a more unpredictable way. In particular, it is not realistic to expect that all the agents get or send their news items simultaneously. But even if the agents acted on news in a sequential way (i.e., instead of processing n pairs of numbers in one step, the agents would average pairs of numbers one after another), the algorithm would still converge to the mean exponentially fast. More precisely, it can be shown that after k iterations of the "averaging operation" the variance drops to $(1 - 1/n)^k$ of its initial value, thus a single cycle (of n iterations) reduces it approximately by factor $e \approx 2.71$. Let us note that the averaging operator does not change the mean.

3.2 Systematic Averaging

The BA algorithm has one drawback: the lack of adaptivity. Sometimes we would like the system to dynamically adjust the output value (in our case: the estimate of the mean) in response to a changing situation: a modification of agents' own values, changes of the number of agents that form the network, temporary faults in communication channels, etc. The systematic averaging algorithm achieves

adaptivity by constantly propagating agents' current values and temporal averages through the news agency. Therefore, any change in the incoming data will quickly affect the final result.

Let us fix a small positive integer d, e.g., $d = 15$, that will control the depth of the propagation process. The SA algorithm works with news items that are vectors of $d + 1$ numbers. The first element of a news item $\mathbf{x}$, x_0, will always be an agent's value (we will call it a 0-order estimate of the mean), x_1 will be the average of two 0-order estimates (we will call it a 1-order estimate), $\ldots$, x_d will be the average of two estimates of order $d - 1$ (and will be called an estimate of order d). In this way consecutive elements of $\mathbf{x}$ will be "balanced": they will be averages of $1, 2, 4, \ldots, 2^d$ of original values. Clearly, the result this propagation is represented by x_d.

The systematic averaging algorithm, when applied to news items a[] and b[] processes the estimates from left to right:

```
NewsItem NewsUpdate({a[], b[]}){
           create a news item c[d];
           c[0]= current value of the agent
           for (i=1; i<=d; i++)
             c[i]+=(a[i-1]+b[i-1])/2;
           return c[]; }
```

Using the same argument as above we can show that the SA algorithm reduces the variance of the input data exponentially fast. Moreover, the system reacts to changes in the input data within d iterations.

3.3 Cumulative Averaging

Both algorithms, BA and SA, reduce variance exponentially fast. Unfortunately, due to randomness that is involved in the sampling mechanism of the newscast engine, the output values might still be different from the true mean. Our third algorithm, cumulative averaging, CA, solves this problem by running two processes in parallel: in one process agents update their local estimates of the mean of the incoming data, in the other one the mean of these estimates is collectively calculated (by the BA procedure). More precisely, news items consist of two numbers: the private value of an agent and the current estimate of the mean. Each agent is counting and summing up all incoming private values (first process) and returning the average of the incoming estimates and its own private value. We will leave further implementation details to the reader. The reader can also verify that local estimates of means tend to the true mean (with the increasing number of cycles), so it is guaranteed that the whole algorithm also converges to it.

3.4 Experiments and Results

For the purpose of simulation we used the actual newscast model instead of the idealized model presented in the introduction. This is very useful in illustrating that the intuitions and the mathematical analysis based on the idealized model

provide a practical approximation when working in the newscast model. To gain experimental data on the behavior of our system we performed runs with various number of agents (10000, 20000, and 50000), and different data sets. For each case we executed 100 independent runs with cache size 20 and terminated after 100 cycles. The data sets included *Gaussian* (where the value of each agent is drawn independently from a Gaussian distribution), *half-half* (where half of the agents hold the value 0, the other half has value 1), and *peak*, where one agent has value being the number of nodes and all other agents have value 0, so the "correct" average is 1.

It turned out that with respect to the convergence rate the BA algorithm was fastest (20-30 iterations were sufficient), the SA algorithm was slower (about 50 iterations were needed) and the slowest was the CA algorithm (about 100 iterations were needed). On the other hand, with respect to accuracy, the situation was opposite: the BA was worst, CA better, and CA the best. The actual deviation from the "true mean" strongly depended on the initial distribution of the data. For example, on the "hardest" peak distribution the average output of the SA algorithm was 0.98, with the standard deviation 0.265, whereas the BA was producing 0.935, with the standard deviation 0.656. A more extensive survey of the results is presented in [7].

4 An Illustrative Example: Naive Bayes

A central problem in data mining is classification: given some records $\mathbf{x}_1, \ldots, \mathbf{x}_r$, represented here by vectors of fixed length p, with their class labels, $y_1, \ldots y_r$, one wants to build a classification procedure that assigns labels to new observations that are not labelled. This classification procedure might have a form of a decision tree, a regression formula, a description of a joint probability distribution, etc., [3]. In this paper we will focus on a very simple, yet powerful, classification procedure called Naive Bayes. Additionally, we will assume that all attributes (vector elements) are discrete and take values in $V = \{v_1, \ldots, v_k\}$; class labels are assumed to belong to $\{c_1, \ldots, c_m\}$.

The Naive Bayes procedure finds $p(y = c_l|\mathbf{x})$, for $l = 1, \ldots, m$ with help of some probability estimates that are easy to find. The class with the highest probability is chosen as the label for $\mathbf{x}$. Indeed, if we assume that attributes are conditionally independent with respect to the class attribute (it is a naive assumption therefore the name: Naive Bayes), the probabilities $p(y = c_l|\mathbf{x})$ can be expressed in terms of $p(x_i = v_j|y = c_l)$ and $p(y = c_l)$, for $i = 1, \ldots, p$, $j = 1, \ldots, k$, and $l = 1, \ldots, m$, where x_i denotes the i-th coordinate of $\mathbf{x}$ (the value of the i-th attribute):

$$p(y = c_l|\mathbf{x})p(\mathbf{x}) = p(y = c_l) \prod_{i,j} p(x_i = v_j|y = c_l).$$

The term $p(\mathbf{x})$ can be eliminated as we know that $\sum_{l=1}^{m} p(y = c_l|\mathbf{x}) = 1$. Clearly, given the data, all the probabilities that we need can be expressed by ratios:

$$p(y = c_l) = \frac{\text{number of observations with label } c_l}{\text{number of all observations}}, \text{and}$$

$$p(x_i = v_j | y = c_l) = \frac{\text{number of observations with label } c_l \text{ s.t. } x_i = v_j}{\text{number of all observations with label } c_l}.$$

Therefore, to implement the Naive Bayes procedure in the newscast model we only have to know how to calculate ratios of some counts. More precisely, let us consider n agents that form a newscast network and let each agent store two numbers a_i and b_i, for $i = 1, \ldots, n$. We are interested in estimating the value of $r = (\sum a_i)/(\sum b_i)$. Once we know how to determine r we know how to calculate all the conditional probabilities we need. Fortunately, the ratio r can be expressed as a combination of two means: $r = (\sum a_i/n)(n/\sum b_i)$. Therefore, any algorithm that was described in the previous section, after a slight modification (we have to estimate several means at the same time), can be immediately used for finding the Naive Bayes classifier for data that is arbitrarily distributed among the agents.

Let us note that most statistics that are used by other classification algorithms are defined in terms of ratios (or probabilities) that have the same form as described above. For example, information gain, gain ratio, Gini index and χ^2 statistics that are used by decision tree inducers: ID3, C4.5, CART and CHAID, respectively, [3]. Consequently they can be implemented within the newscast framework.

5 Summary and Conclusions

The main contribution of this paper is the theoretical and experimental evidence for the feasibility of a novel approach to distributed data mining. The particular type of distributed data mining task we handle constitutes of seeking a model for data spread over a number of sites (here, agents). The challenge is twofold. Firstly, the number of agents can be extremely large (here, up to 50000) and the amount of data per agent can be very small (here, one single value). Secondly, the data might change on-the-fly so the system should be able to adjust the model to these changes automatically. We have reduced the general data mining task to calculating averages demonstrating that it forms the basis for "real" data mining algorithms, such as Naive Bayes or decision trees.

The technical approach we follow is based on the newscast model of computation. We have designed, implemented, and executed algorithms fitting into this model naturally inheriting its main properties: robustness, scalability, and efficiency. For some of these algorithms we have proved theoretical properties on convergence speed and also provided experimental data to show the systems behavior from various perspectives, such as the "averaging power", convergence behavior, and adaptivity in case of changing the data set on-the-fly.

Our current research focuses on the development of other "building blocks" for data mining algorithms, like quantile estimation or various discretization algorithms. We are also experimenting with newscast implementations of incremental algorithms for constructing decision trees.

References

[1] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database management. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Vancouver, Aug. 1987. ACM.

[2] P. T. Eugster, R. Guerraoui, S. B. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probablistic broadcast. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'01)*, Göteborg, Sweden, 2001.

[3] D. Hand, H. Manilla, and P. Smyth. *Principles of Data Mining*. The MIT Press, Cambridge, Massachusetts, London, England, 2001.

[4] M. Jelasity, M. Preuß, M. van Steen, and B. Paechter. Maintaining connectivity in a scalable and robust distributed environment. In H. E. Bal, K.-P. Löhr, and A. Reinefeld, editors, *Proceedings of the Second IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002)*, pages 389–394, Berlin, Germany, 2002. IEEE, IEEE Computer Society.

[5] M. Jelasity and M. van Steen. Large-scale newscast computing on the Internet. Technical Report IR-503, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, Oct. 2002. http://www.cs.vu.nl/globe/techreps.html.

[6] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probablistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 2003. To appear.

[7] W. Kowalczyk, M. Jelasity, and A. Eiben. : Towards data mining in large and fully distributed peer-to-peer overlay networks. Technical Report IR-AI-003, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, May 2003.

[8] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto, 2002.

[9] B. Paechter, T. Bäck, M. Schoenauer, M. Sebag, A. E. Eiben, J. J. Merelo, and T. C. Fogarty. A distributed resource evolutionary algorithm machine (DREAM). In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000)*, pages 951–958. IEEE, IEEE Press, 2000.

[10] B.-H. Park and H. Kargupta. Distributed data mining: Algorithms, systems, and applications. In N. Ye, editor, *The Handbook of Data Mining*. Lawrence Erlbaum Associates, Inc., 2003.

Dynamic vehicle routing using an ABC-algorithm

R. Kroon, L.J.M. Rothkrantz

Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands

Abstract

In the past years the application of agent algorithms based on the natural behaviour of ants have shown to be successful in routing data through communication networks. Using the trail-laying abilities of ants the mobile agents are able to create well performing routing tables. In this paper an Ant Based Control algorithm is applied to the routing of road traffic trough a city. The algorithm is tested in a simulation environment that makes it possible to show the effect in different cities and circumstances. The agents do not move through a real city, but use a model of a city map. This model is supplemented with actual data from the traffic in the city. This enables the agents to divert traffic from congested routes, which improves travelling-times.

1. Introduction

Road traffic is getting busier and busier each year. Everyone is familiar with traffic congestion on highways and in the city. And everyone will admit that it is a problem that affects us both economically as well as mentally. Furthermore finding your way in an unknown city can be very difficult even with a map. Navigation systems like CARiN can help in such cases. These systems display the route to be followed when the user has entered his destination. The latest versions are also able to use congestion information to avoid trouble spots. But such information is only available for highways and not in a city.

This paper addresses the dynamic routing of traffic in a city. We want to set up a routing system for motor vehicles that guides them through the city using the shortest way in time, taking into account the load on the roads. Furthermore we want the routing system to be distributed, for more robustness and load distribution.

The routing system uses a routing algorithm based on earlier versions of Ant Based Control-algorithms. Exact routing algorithms like Dijkstra's algorithm only apply to central routing. And ant-based algorithms have proven to be superior to other distributed routing algorithms in [1,2]. In

[2] an ant-based algorithm was used for routing and load balancing in a telephony network. In [3] the algorithm is applied to packet switched networks with basic ideas taken from [1]. And now we will apply a variant of the algorithm to a traffic network in a city.

2. Ant-based control for network management

We can use the idea of emergent behaviour of natural ants to build routing tables in any network. We will apply it in a traffic network in a city, i.e. the composition of the roads and their intersections. This network is represented by a directed graph. Each node in the graph corresponds to an intersection. The links between them are the roads. Mobile agents, whose behaviour is modelled on the trail-laying abilities of natural ants, replace the ants. The agents move across the network between randomly chosen pairs of nodes. As they move, pheromone is deposited as a function of the time of their journey. That time is influenced by the congestion encountered on their journey. They select their path at each intermediate node according to the distribution of the simulated pheromone at each node. Each node in the network has a probability table for every possible final destination. The tables have entries for each neighbouring node that can be reached via one connecting link. The probabilities influence the agent's selection of the next node in their journey to the destination node. The probability of the agents choosing a certain next node is the same is the probability in the table.

The probability tables only contain local information and no global information on the best routes. Each time an agent visits a node the next step in the route is determined. This process is repeated until the agent reaches its destination. Thus, the entire route from a source node to a destination node is not determined beforehand.

Agents are launched at each node with regular time intervals with a random destination node. They travel around the network using the probabilities in the probability tables. The probabilities per destination are all filled with equal values for all nodes before the process begins.

3. Design

This section explains the design of the routing system.

3.1 Dynamic data

To route the traffic dynamically through a city we need dynamic data about the state of the traffic in the city. This can for example be directly from sensors in the road-surface. Such sensors can count vehicles and measure the speed of the vehicles. That information can be used to compute the time it takes to cover a part of the road. Another source can be the traffic information services. They can inform the system about congestion, diversions of the road, roadblocks and perhaps open bridges. And finally the vehicles themselves can provide the system with information about the path they followed and the time it took them to cover it. The current technology enables to fix the position of a vehicle with an accuracy of a few meters. That position can be communicated to the system along with the covered route.

For our routing system we will at first only use the latter type of information as dynamic data. But of course the model is open for additional types of dynamic data. The information from the vehicles is handled by a separate part of the routing system, called the timetable updating system. This subsystem takes care that the information is processed for use by the ant-based algorithm. This way one vehicle drives a certain route and sends its performance to the routing system. Another vehicle is able to use that information to choose the shortest route.

3.2 Architecture

We will now explain the structure of the system from the viewpoint of the vehicle and its driver. A vehicle is driving through a city and it wants to know the way. The driver enters the address where he wants to go and expects a routing system to tell him where to go. Besides the destination the routing system needs to know the location where the vehicle is at the moment. Therefore the vehicle sends a request to a satellite of the GPS (Global Positioning System). This is shown by arrow A in figure 1. GPS is a system that can determine a position of the sender with an accuracy of a few meters. So the GPS-satellite answers the vehicle with its current position (arrow B). This position is measured in latitude/longitude co-ordinates. In the vehicle these co-ordinates are translated in a position on a certain road with the aid of a digital map of the city. Now the vehicle has enough information to request the routing system what route to follow. The vehicle sends its position and its desired destination along with the request for the route to the routing system (arrow D). Arrow E is the answer from the routing system that contains the route that the vehicle

should follow. These steps are pretty obvious, but we have skipped arrow C. This arrow indicates that the vehicle provides the routing system with information about the route is has followed since the previous time. The information consists of (1) the location and time at the moment of the previous update, (2) the location and time at this moment and (3) the route that the vehicle has followed in between these times and locations. Table 1 shows a detailed enumeration of the information that is send along the indicated arrows.

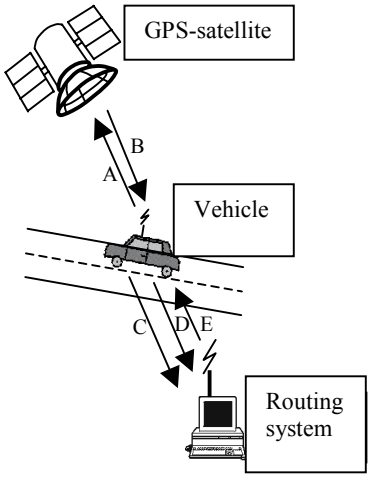


Figure 1:Communication of the vehicle

	From	To	Data
A	Vehicle	GPS-satellite	REQUEST_POSITION
B	GPS-satellite	Vehicle	ANSWER_POSITION, latitude/longitude co-ordinates
C	Vehicle	Routing system	UPDATE, previous time/position, covered road A, covered road B, covered road C, …, current time/position
D	Vehicle	Routing system	REQUEST_ROUTE, current position, destination
E	Routing system	Vehicle	ANSWER_ROUTE, road A, road B, road C, …

Table 1: Communicated data between the different objects

3.3 Routing problem

The most important problem of this research is solved by the *timetable updating system* and the *route finding system*. These two subsystems together form the *routing system*. The function of the *route finding system* will be clear: we are building a system to route vehicles. The reason why we need the *timetable updating system* is the following. The *route finding system* needs information about the state of the network. A static route finding system could use a fixed set of data, but we will use a dynamic route finding system that needs dynamic data. Those data are provided by the *timetable updating system*. That information can be for example the load of the parts of the network but a more direct and therefore more practical type of information is the time it takes to cover a road. Vehicles send information about their covered route to the *timetable updating*

system. From that information this system computes the travelling-times for all roads and stores it in the timetable in the *memory*. Besides the timetable also a history of measurements is stored in the memory. The *route finding system* uses the information in the timetable to compute the shortest routes for the vehicles. When a vehicle requests route information, the *route finding system* sends this information back to the vehicle.

3.3.1 Route finding system

This system uses the earlier mentioned ant-based control algorithm (ABC-algorithm). This algorithm makes use of forward and backward agents. The forward agents collect the data and the backward agents update the corresponding probability tables in the associated direction. The algorithm consists of the following steps:

- At regular time intervals from every network node **s**, a forward agent is launched with a random destination **d**: F_{sd}. This agent has a memory that is updated with new information at every node **k** that it visits. The identifier **k** of the visited node and the time it took the agent to get from the previous node to this node (according to the timetable) is added to the memory. This results in a list of ($\mathbf{k}$, $\mathbf{t}_k$)-pairs in the memory of the agent. Note that the agent can move faster than the time in the timetable.
- Each travelling agent selects the link to the next node using the probabilities in the probability table. The probabilities for the nodes that have already been visited by this agent are filtered out for this agent. Then a copy of the remaining probabilities is made for this agent and these probabilities are normalized to 1. Only this agent uses this temporary probability distribution to choose a next node. So the probability table is not updated yet.
- If an agent has no other option than going back to a previously visited node, the arising cycle is deleted from the memory of the agent.
- When the destination node **d** is reached, the agent F_{sd} generates a backward B_{ds}. The forward agent transfers all its memory to the backward agent and then destroys itself.
- The backward agent travels from destination node **d** to the source node **s** along the same path as the forward agent, but in the opposite direction. It uses its memory instead of the probability tables to find its way.

- The backward agent with previous node **f** updates the probability table in the current node **k**. The probability p_{df} associated with node **f** and destination node **d** is incremented. The other probabilities, associated with the same destination node **d** but another neighbouring node, are decremented. The used formulas are given below.

The probability of the entry corresponding to the node f from which the backward agent has just arrived is increased using the following formula:

$$P_{new,f} = \frac{P_{old,f} + \Delta P}{1 + \Delta P} \tag{3}$$

Here, Pnew,i is the new probability, Pold,i the old probability and ΔP the probability increase. ΔP should be inversely proportional to the age of the forward agent. The formula we use is:

$$\Delta P = \frac{a}{t} + b \tag{4}$$

Where a and b are constants and t is the trip-time of the forward agent from this node to the destination node. This trip-time is the sum of the trip-times from this node to the destination node of the forward agent. We do not take into account that the conditions of the traffic network can change from the moment that the node is visited by the forward agent and the updating of the backward agent.

The other entries in the probability table with the same destination but other neighbouring nodes are decreased using the formula:

$$P_{new,i} = \frac{P_{old,i}}{1 + \Delta P}, \quad \forall\, i \neq f \tag{5}$$

4. Experiments

As a proof of concept we run an experiment in a traffic network as displayed in Figure 4. For vehicles from the road between intersections 5 and 6 or 6 and 7 there is only one reasonable path to their destination. The alternative paths, which go via intersections 3, 4 and 8, will hardly ever be more attractive for those vehicles. For the vehicles from the roads between intersection 1 and 2 and intersection 9 and 10 there are two reasonable options. They can take the northern path via intersections 3, 4 and 8, or they can take the southern path via intersection 6. The length of both routes is different. Taking the length and maximum speed into

account, the path for a vehicle from 1/2 to 9/10 can be covered in 75 seconds when passing intersections 2, 6 and 9. The alternative via intersections 2, 3, 4, 8 and 9 will cost at least 82 seconds. So all vehicles from 1/2 to 9/10 and vice versa will initially take the southern route. But intersection 6 of the southern route is a point where many vehicles from different roads join and cross each other's path. Therefore it is controlled by traffic lights. These traffic lights make the crossing safer and the priority for vehicles from different roads is distributed more fairly. On the other hand the traffic might perceive a considerable delay at this intersection because of the heavy load and the traffic lights. This will cause the northern path to be more attractive for vehicles from intersection 1/2 to 9/10 and vice versa. So we expect the vehicles that follow the advice of the Routing system to drive via the northern roads, where there are no delaying intersections. This should result in faster routes for these vehicles as opposed to the vehicles that do not use the Routing system.

4.1.1 Results

The first run of this experiment provided the following graphs (Figure 2 and Figure 3). These graphs show the differences in the average travel time of standard and smart vehicles over period of 40 minutes (2400 seconds). The standard vehicles do not use the Routing system, the smart vehicles do. The value at the end, after 2400 seconds, is the average of all measured travel times since the start of the experiment until the end.

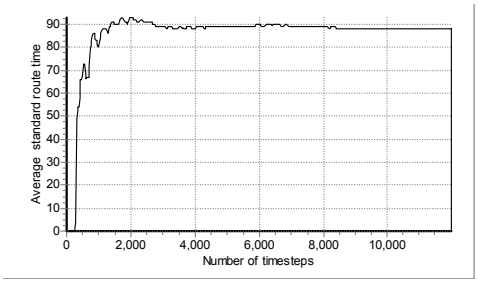


Figure 2: Average standard route time

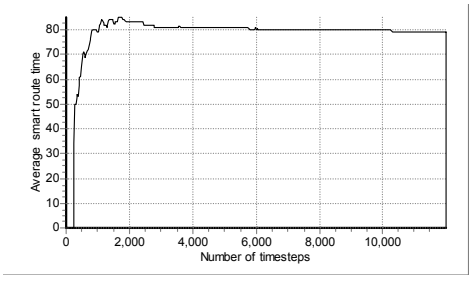


Figure 3 Average smart route time

When we zoom in on the graphs (Figure 2 and Figure 3) we see that the (rounded) value for the standard vehicles is 88 seconds and the value for the smart vehicles is 79 seconds. So the overall profit for the smart

vehicles is 10 % on average as opposed to the standard vehicles, which do not use the Routing system.

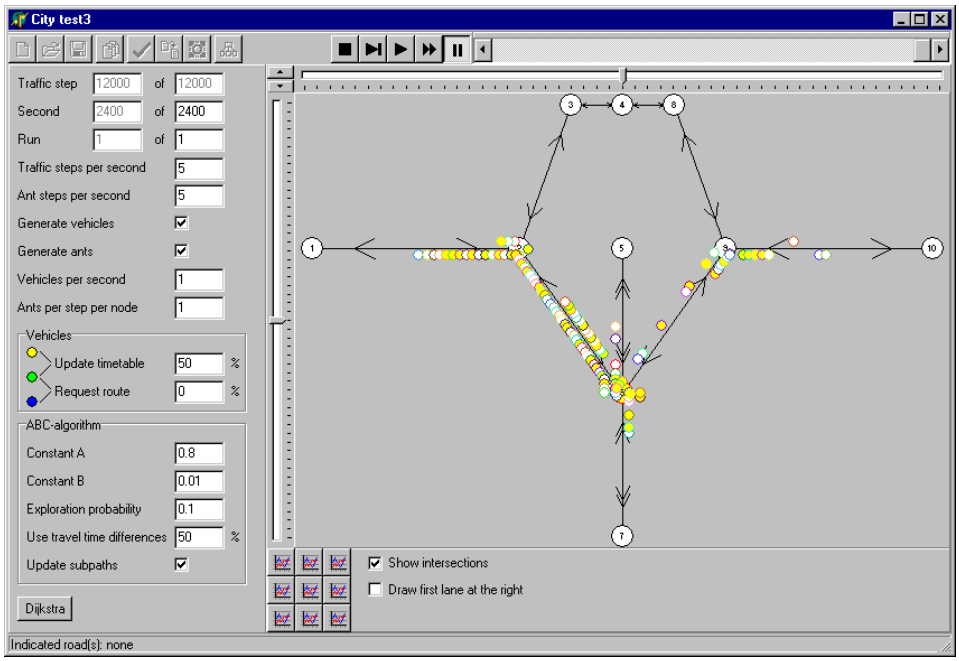


Figure 4: Screenshot of the City program after 2400 second without Routing system

References

[1] G. Di Caro and M. Dorigo. *AntNet: distributed stigmergetic control for communication networks*. Journal of Artificial Intelligence Research (JAIR), 9, 317-365, 1998

[2] R. Schoonderwoerd, O. Holland, J. Bruten, L.J.M. Rothkrantz. *Load balancing in telecommunication networks*, Adaptive Behaviour, 5, 2 , 1997.

[3] L.J.M. Rothkrantz, J.C. Wojdel, A. Wojdel, H. Knibbe. *Ant based routing algorithms*, Neural Network World, Vol. 10, No.3, July 2000

[4] H. Dibowski, L.J.M. Rothkrantz *Hierarchical routing system using Ant Based Control*, ITS-report, July 2003

Situation recognition as a step to an intelligent situation-aware crew assistant system

Quint Mouthaan Patrick Ehlert Leon Rothkrantz

Delft University of Technology, Mekelweg 4, 2628 CD, Delft

Abstract

In this paper we present a system that can recognize situations during a flight in real-time based on data from simulated aircraft systems. The system uses Bayesian belief networks to calculate the probabilities of both the start and end of all possible situations and from this distillates the most probable situation. The situation recognizer system is part of our test environment to create better human-machine interfaces in the cockpit.

1. Introduction

Anyone who has seen the cockpit of an F-16 aircraft knows that it is stuffed with control buttons, meters, and displays, providing the pilot with a wealth of information. Since the F-16 is capable of speeds of over 2000 Km/h, pilots have very little time to process the large amount of available information and make decisions. To help a pilot deal with information processing and decision-making and avoid information overload, an intelligent pilot-vehicle interface or Crew Assistant System (CAS) or has been proposed [1,2]. A typical CAS is shown in Figure 1. The idea is that the system presents relevant information to the pilot at the right moment and in the appropriate format, depending on the situation, the status of the aircraft, and the workload of the pilot. It is even possible that the CAS takes over (simple) tasks. Not only military pilots can benefit from such a system, it is useful for commercial pilots as well.

The Intelligent Cockpit Environment (ICE) project is a project of the Knowledge Based Systems group of Delft University of Technology. The goal of the ICE project is to design, test, and evaluate computational techniques that can be used in the development of intelligent situation-aware CASs. Using methods from artificial intelligence, ICE focuses primarily on the data fusion and reasoning part of these systems. Special issues addressed in the ICE project are situation recognition, mission or flight plan monitoring, pilot workload monitoring, and attack management [3,4].

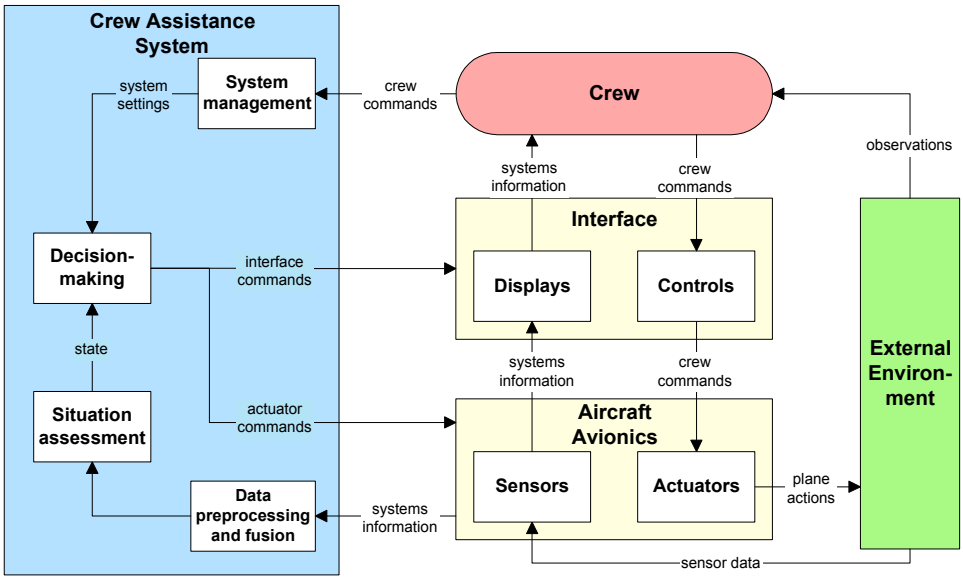


Figure 1: a generic crew assistance system architecture

In this paper we present a system for real-time situation recognition, which is an important subsystem for situation assessment in any CAS. First, we will describe the design of the recognition system. Then we will discuss an example scenario that was used to test our system. Finally, we will draw some conclusions about the performance of the system.

2. The design

The goal of our system is to derive the current situation in real-time from available aircraft data. The system receives information from a simulator about the state of the airplane (e.g. airspeed, altitude, pitch), the actions of the pilot (e.g. lowering the landing gear, changing display settings), and the environment (e.g. other planes, or a missile that has been launched). Our system will use all this information to determine which situation is occurring. For every situation, the actions the pilot is expected to perform and typical situation-related events are defined. These events can either be changes in the state of the airplane or changes in the environment. During a flight the system will compare the received information with the stored situations data and it will try to determine which situation is occurring.

2.1. System architecture

The architecture of the situation recognition system is shown in Figure 2.

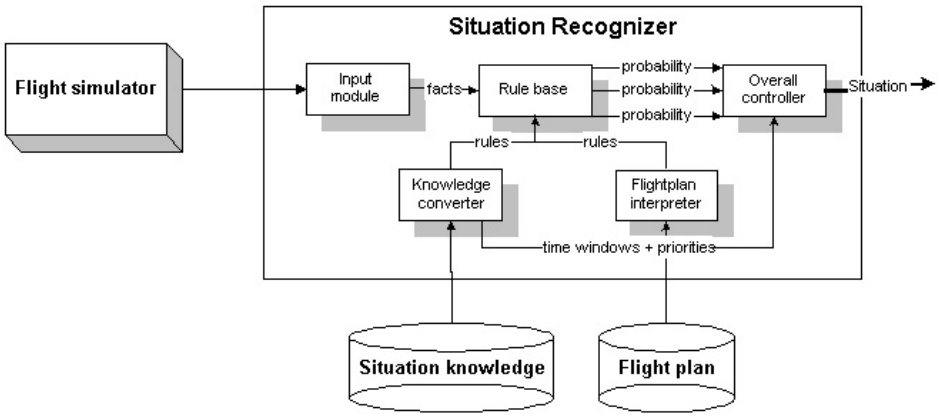


Figure 2: the architecture of the situation recognition system

The input module receives aircraft data from the flight simulator and converts this data to facts that are forwarded to the rule base.

The knowledge converter converts all the situations knowledge that is stored in an XML file to IF-THEN rules and puts them in the rule base.

The flight plan interpreter converts the information in the flight plan to a number of rules that are put in the rule base. These are rules that predict which situations will occur in the near future.

The rule base contains all the rules that have been generated by the earlier described modules. When data from the flight simulator is added to the rule base, some of the rules will fire and generate probabilities concerning the start or end of a situation that are passed to the overall controller.

The overall controller receives situation probabilities from the rule base as well as some extra information about the situations. The overall controller combines the probabilities and calculates for every situation the probability that it has started and the probability that it has ended. It then draws a conclusion about the situation that is most likely to be the current one. Calculating probabilities is done using Bayesian Belief Networks (BBNs), which will be discussed in the next section.

2.2. Probability inference using Bayesian belief networks

We want to calculate two probabilities for every situation: the probability that the situation has started and that it has ended. As a first order approach we have created two BBNs.

2.2.1. The start probability calculator

Figure 3 shows the BBN that is used to calculate the probability that a situation has started.

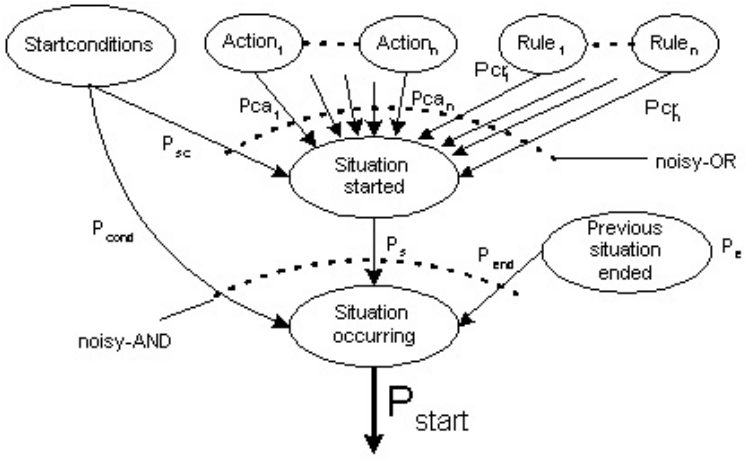


Figure 3: BBN that calculates the probability that a situation has started

The start conditions for a situation are conditions that must be satisfied before a situation can possibly have started. When the start conditions are satisfied, the probability of the start constraints that is specified in the situations knowledge will be the output of this node.

The action probabilities are passed to the BBN by the rules in the rule base when the pilot performs a particular situation-related action. These probabilities all contribute to the probability that the situation is occurring (has been started).

The additional rules are rules that fire when the state of the aircraft changes or when a specific event happens. When they fire they can generate a probability that a situation has started or ended.

The probability calculator (situation started) combines the probabilities of the nodes that have been described above using the noisy-OR model.

The previous situation influences the start probability of a situation because the probability that a situation is occurring should rise when the probability increases that one of the previous situations that can lead to this situation has ended.

Based on this BBN the probability that a situation has started and is occurring can be calculated with the following formula:

$$P_{start} = P_{cond} * P_{end} * P_s$$

$$P_{cond} = \begin{cases} 0 & if\ P_{sc} = 0 \\ 1 & if\ P_{sc} > 0 \end{cases}$$

$$P_s = 1 - ((1 - P_{sc}) * \prod_{i=1}^{n} (1 - Pca_i) * \prod_{j=1}^{n} (1 - Pcr_j))$$

In this formula, P_{sc} is the probability of the start conditions, P_{end} is the probability that one of the previous situations has ended, Pca_i is the probability of the i-th action that should be performed during the situation and Pcr_j is the probability of the j-th event or change in state that can occur during the situation.

2.2.2. The end probability calculator

In Figure 4 the BBN is shown that calculates the probability that the situation has ended. In this BBN we see a lot of the same nodes as in the belief network for the start of the situation. The nodes that are different are discussed below.

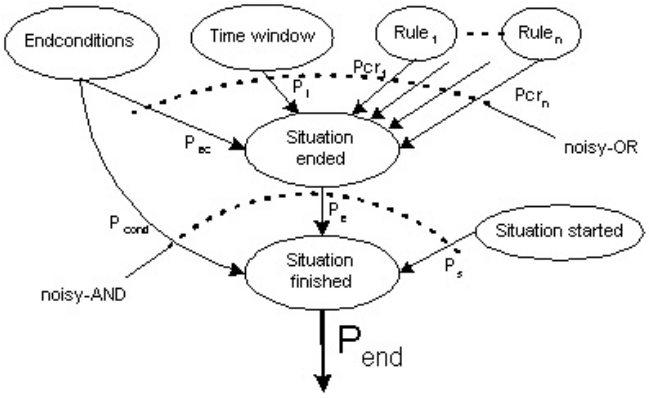


Figure 4: BBN that calculates the probability that a situation has ended

The time window for a situation is the maximum duration of that situation. If the start of a situation has been detected the probability that it has ended should grow after a certain time.

The situation started node produces a 1 if the situation has started and a 0 if the situation has not yet started. This node is necessary because we only want to calculate the probability that the situation has ended, after (a probable) start of that situation.

The probability that the situation is ended can be calculated with the following formula:

$$P_{end} = 1 - ((1 - P_{sc}) * (1 - P_t) * \prod_{i=1}^{n} (1 - Pcr_i))$$

More information about the design of our system can be found in [5].

3. Experiments and results

The system was tested using Microsoft's Flight Simulator 2002. Experiments were performed with an F-16 and with a Cessna airplane. The results of one of our experiments are presented in Table 1. The first column contains the names of the situations (19 different situation were defined) that occurred and/or were detected. The second column contains the times at which we considered the situations to be started. The third column contains the times at which the situations were detected by the system. The times are given in seconds from the moment the program was started. The particular mission of Table 1 consisted of an attack on a ground target in an F-16. During the flight to the target the pilot had to check his course twice (navigating). After the attack had been performed the pilot returned to the airbase and landed the airplane.

Table 1: results of an experiment flight

Situation	Time started (s)	Time detected (s)
Startup	0	0
Taxiing to runway	10	12
Taking off	14	15
Normal flight	43	34
Navigating	83	83
Normal flight	91	91
Navigating	123	123
Normal flight	128	128
Visual attack	186	193
Normal flight	221	221

Landing	361	366
Aborting a landing	-	410
Taxiing from runway	409	410
Shutdown	427	427
Error rate = 0.06 (26 seconds)		

From the table it is clear that the program is able to recognize most situations in a matter of seconds. The error rate was calculated by calculating the amount of time that the recogniser was incorrect. The program has some difficulty in detecting the situation "normal flight" after "taking off" (this problem occurred in other experiments as well and has to be looked into). The landing was a normal landing, but as is shown in the table the program thought for a moment that the landing was being aborted. This happened when the program knew that the landing had been finished and looked for the situation with the highest start probability. This turned out to be the situation *Aborting landing*. This is because at some point the pilot had moved the throttle to the maximum. This action was still in the memory of the program when the landing ended. The fact that the landing gear was raised at the start of the landing was also still in the memory of the program. Because of this the probability that the landing was being aborted was high and the situation *Aborting landing* became the current one once the landing had finished. However as soon as that happened the program realized that the airplane was actually taxiing and it corrected the mistake immediately.

Our other experiments showed similar results. On average the error rate over the performed experiments (4 flights) was 0.08, with 0.05 being the lowest recorded error rate and 0.11 being the highest.

4. Conclusions and future work

We have created a system that can recognize the current situation during a flight with an F16 and with a Cessna. The system is based on a probabilistic model. A rule base was created that compares data from a flight simulator with the situations knowledge defined in an XML file. The rules in the rule base generate a number of probabilities that are combined using BBNs to calculate the probabilities that the situations are occurring. Based on these probabilities a conclusion is drawn about the situation that is most likely to be occurring.

We did not prove the correctness of the system, but investigating a number of test scenarios, the system seems to works fairly well. It makes

few mistakes and is able to correct them. Furthermore it is able to come to a conclusion about the current situation in real-time.

Future work will consist of solving the "Taking-off/Normal flight"-transition problem, adjusting the timeframe a particular action or event is stored, and adding causality to make the system more reliable. In addition we would like to improve the system to include more and synchronous situations and compare its results to other approaches such as Dynamic network models or production systems. We also plan to use the system in conjunction with a workload assessment module that is under development to construct a more complete situation awareness module. This situation awareness module can be used in an intelligent cockpit system that monitors and supports the pilot during a flight.

References

[1] Banks, S.B and Lizza, C.S. (1991) "Pilot's Associate: a cooperative, knowledge-based system application", in *IEEE Intelligent Systems, Vol. 6*, No. 3, pp. 18-29, June 1991.

[2] Onken, R. (1997) "*The cockpit assistant system CASSY as an on-board player in the ATM environment*", paper presented at 1st USA/Europe Air Traffic Management R&D Seminar, Saclay, France, June 1997.

[3] Ehlert, P.A.M. and Rothkrantz, L.J.M. "*The Intelligent Cockpit Environment Project*", Research Report DKS03-04/ICE 04, Knowledge Based Systems, Delft University of Technology, The Netherlands.

[4] http://www.kbs.twi.tudelft.nl/Research/Projects/ICE/

[5] Mouthaan, Q.M. (2003) "*Towards an intelligent cockpit environment: a probabilistic approach to situation recognition in an F-16*", MSc. thesis, Knowledge Based Systems, Delft University of Technology, The Netherlands.

Proper Refinement of Datalog Clauses using Primary Keys

Siegfried Nijssen Joost N. Kok

LIACS, Leiden University,
Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands
{snijssen,joost}@liacs.nl

Abstract

Inductive Logic Programming (ILP) is frequently used to data mine in multi-relational databases. However, most ILP algorithms disregard primary key information which is often available for such databases. This work demonstrates several disadvantages of the *mode refinement* operator that has been used in many multi-relational data mining algorithms in combination with both traditional subsumption and subsumption under Object Identity. We show how primary key information can be incorporated in this refinement operator and provide evidence that the resulting operator has several desirable properties in comparison with the traditional approaches. Especially, we will show that our refinement operator is *proper*.

1 Introduction

In multi-relational data mining research, much attention has been given to Inductive Logic Programming (ILP). A multi-relational database can be mapped to a Datalog database straightforwardly: relations are mapped to predicates and attributes to predicate arguments. ILP algorithms can be applied subsequently.

In this database-to-Datalog mapping, some information about databases is often disregarded: for most databases also *primary keys* and *foreign keys* of relations are defined. These keys describe some restrictions on a database. Our observation is that it is useless to query databases for information that cannot be stored according to the primary key information. Primary keys should therefore also be used to restrict the queries that an ILP algorithm considers. We will formalize this by defining a *downward refinement operator* which uses primary keys.

Every ILP algorithm traverses a search space of clauses of a certain language in some structured way using a so-called *refinement operator*. A *downward* refinement operator ρ is an operator that creates more specific clauses starting from very general clauses. Given a language of clauses and a *quasi-order* on these clauses, several desirable properties for refinement operators have been identified [6]:

(1) ρ should be locally finite: all refinements of a clause should be computable within finite time;

(2) ρ should be complete: given a clause, every clause in the language which is more specific according to the quasi-order should be obtainable by (repeatedly) applying the refinement operator;

(3) ρ should be proper: after refinement, according to the quasi-order the refined clause should always be more specific than the original clause (and therefore never equivalent).

If refinement operator ρ satisfies these properties, then this operator is called *ideal*.

In most ILP systems, the concepts of "more specific" and "general" are modeled using a quasi-order called θ-subsumption. This choice has a major drawback: if a refinement operator is finite and complete under θ-subsumption, it can be shown that this operator can never be proper; as a result, it is possible that a clause is infinitely refined without obtaining a more specific clause.

To face this problem, in [3] a different quasi-order based on subsumption was defined: subsumption under Object Identity. Under Object Identity a clause is evaluated in a different, more restricted way than is usual. Ideal refinement is possible under Object Identity. The additional restrictions however appear to be undesirable in many situations.

In this work, we will concentrate on a special downward refinement algorithm: refinement using *modes*. Mode refinement has been applied in several data mining algorithms [1, 2, 4, 5, 7, 8], and has shown its usefulness in these publications.

Our paper is organized as follows. In the second section, we will review both traditional subsumption and the concept of Object Identity, and we will introduce mode refinement. With several examples we will illustrate the problems which occur when either traditional or OI subsumption is used.

In the third section, we will introduce a new quasi-order and an enhanced mode refinement algorithm. As our new way of evaluating clauses is somewhere in the middle between evaluation under Object Identity and ordinary clause evaluation, we call our evaluation technique evaluation under *weak Object Identity*. Within our setup, the weak-OI quasi-order has several parameters, among which the primary keys. We will show that these parameters can be tuned in such a way that our quasi-order reduces to full Object Identity; in this way, our setup is a generalization of full OI. Using examples, we will show that one can also provide parameters for the refinement algorithm such that the resulting clauses do not suffer from the restrictions of full OI. Still, we will provide evidence that this refinement algorithm has exactly the same desirable properties as mode refinement using full Object Identity; more precisely, the refinement algorithm is finite and proper. Section four concludes.

2 Prerequisites and problem description

We will briefly review some terminology [6]. A (Datalog) *atom* $p(t_1, \ldots, t_n)$ consists of a relation symbol p of arity n followed by n terms t_i. A *term* is either a constant or a variable. A *substitution* θ is a set of the form $\{v_1/t_1, \ldots, v_n/t_n\}$ where v_i is a variable and t_i is a term. One can *apply a substitution* θ to an expression e,

yielding the expression $e\theta$, by simultaneously replacing all variables v_i by their corresponding terms t_i. An *atom set* is an unordered set of atoms; an ordered set of atoms is an *atom list*. A *clause* is an expression of the form $h \leftarrow S$, where h is an atom and S is an atom set. In this paper, without loss of generality, we consider the head h of clauses to be a fixed atom; we only consider the bodies of clauses. Previously, two kinds of subsumption have been defined:

- *Traditional θ-subsumption*: an atom set S_1 θ-subsumes an atom set S_2 ($S_2 \succeq S_1$) if there exists a substitution θ such that $S_1\theta \subseteq S_2$.
- *OI-subsumption*: an atom set S_1 OI-subsumes an atom set S_2 ($S_2 \succeq_{OI} S_1$) if there exists an injective substitution θ such that $S_1\theta \subseteq S_2$ and θ does not map any variable to a constant or variable already occurring in S_1.

Under traditional θ-subsumption, two atom sets S_1 and S_2 are considered to be equivalent (denoted by $S_1 \sim S_2$) iff $S_1 \succeq S_2$ and $S_2 \succeq S_1$. This is reasonable as one can show that: $(\forall S'(S' \succeq S_1 \rightarrow S' \succeq S_2) \wedge \forall S'(S' \succeq S_1 \rightarrow S' \succeq S_2)) \Leftrightarrow S_1 \sim S_2$; or, in words: if every possible set of atoms either subsumes two atom sets, or does not subsume any of these two, these atom sets are equivalent and must subsume each other.

We will illustrate these subsumption operators using predicates that encode directed, edge labeled graphs. The assumption is that we are interested in clauses with predicates $e(G, V_1, V_2, L)$ (which encodes that there is an edge from vertex V_1 to a vertex V_2 with label L in graph G) and $is(L, K)$ (which encodes that a label L is a label in the class K). So, our language consists of the set of predicates $\{e/4, is/2\}$; furthermore, we assume the set of constants $\{a, b\}$. The following clauses can be expressed in this language:

$$C_1 = p(G) \leftarrow e(G, V_1, V_2, L_1), is(L_1, a), \quad (1)$$

$$C_2 = p(G) \leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_3, V_4, L_2), \quad (2)$$

$$C_3 = p(G) \leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_3, V_4, L_2), e(G, V_4, V_5, L_3), \quad (3)$$

$$C_4 = p(G) \leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_4, V_5, L_3). \quad (4)$$

Clause C_1 states that a graph contains an edge of class a. Clause C_3 states that a graph contains an edge of class a and furthermore contains a vertex with at least one incoming and one outgoing edge, independent of the label. Under traditional subsumption, $C_1 \sim C_2 \sim C_4$ and $C_3 \succ C_1$.

We will now introduce the *bias* of the (traditional) mode refinement algorithm.

Definition 2.1 *A bias $\mathcal{B}$ is a tuple $(\mathcal{T}, \mathcal{C}, \mathcal{P}, \mathcal{M}, h)$, where $\mathcal{T}$ is a finite set of type symbols, $\mathcal{C}$ is a function that defines a finite set of constants for each type in $\mathcal{T}$; $\mathcal{P}$ is a finite set of declarations of the form $p(T_1, \ldots, T_n)$, where each $T_i \in \mathcal{T}$ and a predicate p occurs at most once. Set $\mathcal{M}$ defines* mode declarations*, which are declarations of the form $p(c_1, \ldots, c_n)$ and consist of a predicate symbol p with arguments c_i, each of which is either '+' (input), '-' (output) or '#' (constant). Set $\mathcal{M}$ may contain multiple modes for the same predicate symbol. h is an atom.*

Note that we use types in our bias; this is not common practice in most publications. The mode refinement algorithm and the bias define a search space of clauses, as follows.

Definition 2.2 *Given a bias $\mathcal{B}$ the mode refinement operator ρ recursively defines a search space $\mathcal{L}(\mathcal{B})$ as follows:*

- *'$h \leftarrow$' $\in \mathcal{L}(\mathcal{B})$;*
- *if $C =$'$h \leftarrow S$' $\in \mathcal{L}(\mathcal{B})$, then $\rho(C) \ni C' =$'$h \leftarrow S, A$' $\in \mathcal{L}(\mathcal{B})$, with $A = p(t_1, \ldots, t_n)$, iff there is a mode $M = p(c_1, \ldots, c_n) \in \mathcal{M}$ such that for every $1 \leq i \leq n$:*
  - *t_i is a variable in $var(C, T_i)$ and $c_i =$'+', or*
  - *t_i is a variable not in $\cup_j var(C, T_j)$ and $c_i =$'-', or*
  - *t_i is a constant in $\mathcal{C}(T_i)$ and $c_i =$'#'.*

  Here, T_i is the type of argument position i, as given by $\mathcal{P}$; $var(C, T)$ is the set of variables in C which occur at argument positions of type T.

An example bias is $\mathcal{B} = (\{G, V, L, K\}, \{K \rightarrow \{a, b\}\}, \{p(G), e(G, V, V, L), is(L, K)\}, \{e(+, -, -, -), e(+, +, -, -), is(+, \#)\}, p(G))$, which encodes a search space of labeled *forests*. One can show that $\{C_1, C_2, C_3, C_4\} \subseteq \mathcal{L}(\mathcal{B})$.

It is clear that this refinement algorithm does not generate all clauses that can be expressed using the given predicates and constants. Using traditional subsumption as quasi-order, one can show that the operator is complete within the sublanguage $\mathcal{L}(\mathcal{B})$. As an example, consider clause $C_3\{V_2/V_3\}$, which is a specialization of C_3. An equivalent clause, $C_3 \cup (C_3\{V_i/X_i | i \neq 2\}\{V_2/X_3\})$, can be constructed from C_3, where X_i are variables not occurring in C_3.

It is clear that for traditional subsumption, mode refinement is not proper either. By adding new atoms in two steps, C_3 can be obtained from C_1. In whatever order the last two atoms of C_3 are added, however, each intermediate clause is equivalent with C_1: $C_2 \sim C_1$ and $C_4 \sim C_1$. If one would decide not to allow a refinement from C_1 to C_2 or C_3, the operator would not be complete: one can show that C_1 cannot be refined to C_4 in that case.

If one applies OI-subsumption as quasi-order, the relations between clauses are different: $C_3 \succ_{OI} C_2 \succ_{OI} C_1$ and $C_3 \succ_{OI} C_4 \succ_{OI} C_1$. For example, to C_2 one may not apply $\theta = \{V_3/V_1, V_4/V_2, L_2/L_1\}$ to obtain C_1, as it maps variables to variables already occurring in C_2.

Under OI, mode refinement is always proper. This follows from the observation that under OI sets of atoms are always reduced [3]. Mode refinement is not complete. With the example bias, clause C_1 cannot be refined into $C_1 \cup \{e(G, V_3, V_1, L_2)\} \in \mathcal{L}(\mathcal{B})$.

A different way of defining OI is to define it using traditional θ-subsumption. We will follow this approach in this paper. Given a set of atoms S, we define $constr(S)$ to be the set of atoms

$$constr(S) = \{(t_1 \neq t_2) | t_1 \neq t_2, t_1, t_2 \in terms(S)\},$$

where $\neq$ is a binary predicate denoted in infix notation, and $terms(S)$ is the set of all terms occurring in atom set S. For example:

$$constr(\{is(L_1, a), is(L_1, K_1)\}) = \\ \{(L_1 \neq a), (L_1 \neq K_1), (a \neq L_1), (a \neq K_1), (K_1 \neq L_1), (K_1 \neq a)\}.$$

The OI-subsumption can then equivalently be defined as:

$$S_1 \succ_{OI} S_2 \Leftrightarrow S_1 \cup constr(S_1) \succ S_2 \cup constr(S_2).$$

When evaluating C_3, it is clear now that $\{(V_1 \neq V_2), (V_2 \neq V_3), (V_3 \neq V_4), (V_4 \neq V_5)\} \subset constr(C_3)$ and $\{(L_1 \neq L_2), (L_2 \neq L_3)\} \subset constr(C_3)$: the nodes must be different, and also all labels must be different.

Assume now that one still wishes to find a theory for predicate p that allows nodes to be equal, then this theory should contain several clauses under OI:

$$\begin{aligned} p(G) &\leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_3, V_4, L_2), && (5)\\ p(G) &\leftarrow e(G, V_1, V_1, L_1), is(L_1, a), e(G, V_3, V_4, L_2), && (6)\\ p(G) &\leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_1, V_4, L_2), && (7)\\ p(G) &\leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_3, V_1, L_2), && (8)\\ &\vdots \end{aligned}$$

for a total of 15 clauses, each of which reflects some case of variable equality. For C_3 even 52 clauses are required. One can show that the number of clauses grows exponentially in the number of variables. For theories in which one would like to allow equality, Object Identity can therefore be very impractical.

In some situations, there are ad-hoc solutions to solve problems caused by OI. Assume that one would like to express the following theory with only one clause:

$$\begin{aligned} p(G) &\leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_2, V_3, L_2), && (9)\\ p(G) &\leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_2, V_3, L_1), && (10) \end{aligned}$$

then one could choose to use another predicate language. Consider a language with predicate $e/4$ and a predicate $ea(G, V_1, V_2)$ which is defined in terms of e and is to express that there is an edge from V_1 to V_2 in label class a. The following clause can then be expressed:

$$p(G) \leftarrow ea(G, V_1, V_2), e(G, V_2, V_3, L_2);$$

in this clause L_2 can be the same label as the label between nodes V_1 and V_2. However, this representation has an unwanted side effect:

$$p(G) \leftarrow ea(G, V_1, V_2), e(G, V_1, V_2, L_1);$$

according to OI-subsumption, this clause is not equivalent to any smaller clause, but by the definition of ea we know that the last atom can be removed. We believe therefore that this construction is undesirable too.

From our point of view, the best solution would be to force Object Identity constraints only to some variables in a clause. The question is how this can be done without loosing the desirable, ideal properties of OI. In the next section we will provide an answer to this question.

3 Weak Object Identity using Primary Keys

We will first formally define the *bias* of a language with primary keys. Immediately after the definitions, we will illustrate their meaning using examples.

Definition 3.1 *A bias with primary keys $\mathcal{B}_K$ is a tuple $(\mathcal{T}, \mathcal{C}, \mathcal{P}, \mathcal{M}, h, \mathcal{K}, OI)$, where $\mathcal{B} = (\mathcal{T}, \mathcal{C}, \mathcal{P}, \mathcal{M}, h)$ is a simple bias as given in Definition 2.1 and $\mathcal{K}$ is a function which defines a set of primary keys for each predicate $p \in \mathcal{P}$. A primary key is a subset of $\{1, \ldots, arity(p)\}$. Set OI is a subset of the types, $OI \subseteq \mathcal{T}$.*

Definition 3.2 *An atom set S is* constrained by a primary key $K \in \mathcal{K}(p)$ *iff:*

$$\forall A_1 = p(t_{11}, \ldots, t_{1n}), A_2 = p(t_{21}, \ldots, t_{2n}) \in S : (\forall i \in K : t_{1i} = t_{2i}) \Rightarrow A_1 = A_2.$$

Definition 3.3 *A clause $C =$ '$h \leftarrow S$' is part of the language $\mathcal{L}_K(\mathcal{B}_K)$ defined by a bias $\mathcal{B}_K$ iff:*

- *$C \in \mathcal{L}(\mathcal{B})$, where $\mathcal{B}$ is the simple part of $\mathcal{B}_K$ and $\mathcal{L}(\mathcal{B})$ is defined according to Definition 2.2.*
- *S is constrained by each primary key in $\mathcal{K}(p) \cup K_t(p)$, for every predicate p. With $K_t(p)$ we denote the trivial key of a predicate p, $\{1, \ldots, arity(p)\}$.*

Furthermore C' is a key mode refinement of C, denoted by $C' \in \rho_K(C)$ (for $C \in \mathcal{L}_K(\mathcal{B}_K)$), iff $C' \in \rho(C)$ and C' is constrained by every primary key.

We will continue with our graph example (not restricted to trees). Assume that we know that in the database under consideration between each pair of nodes there is at most one edge in each direction, and that an edge always has exactly one label, then we can express this knowledge using a primary key:

$$\mathcal{K}(e) \rightarrow \{\{1, 2, 3\}\},$$

as this states that an edge can be identified uniquely by giving a graph and two vertices. If this primary key is part of a bias $\mathcal{B}_K$, then $\mathcal{L}(\mathcal{B}_K)$ does not contain the following clause in any case:

$$p(G) \leftarrow e(G, V_1, V_2, a), e(G, V_1, V_2, b);$$

this expression can never be true given our knowledge about the graph data. As we believe that for most relational databases primary key information is available, we believe that this restriction of a full clausal language is important. Because it reduces the number of clauses that an Inductive Logic Programming algorithm has to consider, we believe that this strategy could yield significant efficiency improvements in many algorithms.

Definition 3.4 *Given two clauses $C_1 =$ '$h_1 \leftarrow S_1$' $\in \mathcal{L}_K(\mathcal{B}_K)$, $C_2 =$'$h_2 \leftarrow S_2$' $\in \mathcal{L}_K(\mathcal{B}_K)$, S_1 $\mathcal{B}_K$-OI-subsumes S_2, denoted by $S_2 \succeq_{\mathcal{B}_K - OI} S_1$, iff $S_2 \cup constr_{\mathcal{B}_K}(S_2) \succeq S_1 \cup constr_{\mathcal{B}_K}(S_1)$, where $constr_{\mathcal{B}_K}(S) = \{(t_1 \neq t_2) | t_1, t_2 \in$ OI-terms$_{\mathcal{B}_K}(S), t_1 \neq t_2\}$ and OI-terms$_{\mathcal{B}_K}(S)$ is the set of terms occurring in S at argument positions i of predicates p for which $T_i \in OI \in \mathcal{B}_K$.*

The main difference with traditional OI-subsumption is that using types, OI constraints are only forced to some variables in a clause. As an example, consider the bias $\mathcal{B}_K = (\{G, V, L, K\}, \{K \rightarrow \{a, b\}\}, \{p(G), e(G, V, V, L), is(L, K)\}, \{e(+, -, -, -), e(+, +, -, -), e(+, -, +, -), e(+, -, -, +), is(+, \#)\}, p(G), \{e \rightarrow \{\{1, 2, 3\}\}\}, \{G, V\})$. The following clauses are part of $\mathcal{L}_K(\mathcal{B}_K)$:

$$C_2 = p(G) \leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_3, V_4, L_2), \quad (11)$$

$$C_5 = p(G) \leftarrow e(G, V_1, V_2, L_1), is(L_1, a), e(G, V_3, V_4, L_1), \quad (12)$$

then $C_5 \succ_{\mathcal{B}_K - OI} C_2$ while $C_5 \not\succ_{OI} C_2$. In comparison with traditional OI, $(L_1 \neq L_2) \notin constr_{\mathcal{B}_K}(C_2)$.

A special case arises when $OI = \mathcal{T}$ and $\mathcal{K} = \emptyset$: all types are subject to Object Identity constraints such that our weak subsumption operator reduces to full Object Identity. Furthermore, by the absence of keys, no key constraints are effective. One can easily see that our refinement operator reduces to a traditional mode refinement operator with traditional OI as quasi-order. This shows that our formalism is a generalization of full OI.

Now temporarily assume that $is(+, -)$ would be part of $\mathcal{M}$ in $\mathcal{B}_K$. The following clauses would both be part of $\mathcal{L}_K(\mathcal{B}_K)$:

$$C_6 = p(G) \leftarrow e(G, V_1, V_2, L_1), is(L_1, K_1), \quad (13)$$

$$C_7 = p(G) \leftarrow e(G, V_1, V_2, L_1), is(L_1, K_1), is(L_1, K_2); \quad (14)$$

however, $C_6 \sim_{\mathcal{B}_K - OI} C_7$; our mode refinement operator would not be proper. If we however assume that $\{is \rightarrow \{\{1\}\}\} \subseteq \mathcal{K} \in \mathcal{B}_K$, then $C_7 \notin \mathcal{L}_K(\mathcal{B}_K)$, as the unproper refinement is not allowed.

Theorem 3.5 *Given a bias $\mathcal{B}_K$, mode refinement as given in Definition 3.3 is proper if for every $M = p(c_1, \ldots, c_n) \in \mathcal{M} \in \mathcal{B}_K$ there is a key $K = \{i_1, \ldots, i_n\} \in \mathcal{K}(p) \cup K_t(p)$ such that for every $i_j \in K$ one of the following holds:*

- *T_{i_j}, the type of the i_jth argument, is included into $OI \in \mathcal{B}_K$;*
- *$c_{i_j} =$'+' or $c_{i_j} =$'#'.*

Proof Outline We will provide a proof by contradiction. Assume that clause $C \sim_{\mathcal{B}_K - OI} C' \in \rho_K(C)$ and (without loss of generality) that C is not equivalent with any smaller clause. In this case, there is a weak OI substitution θ which maps one atom in C' onto another atom in C', resulting in C. Note that $|C'| = |C| + 1$; we may therefore assume that θ only affects one atom $A \in C'$. As a clause must be key constrained, at least one term in every primary key of this atom is different from the corresponding term in $A\theta$. As θ substitutes a variable with a term already occurring in C', each such different term must be a variable of a type not in $OI \in \mathcal{B}_K$. We may therefore conclude that a variable not in OI is part of every key, and that A is the only atom in which these variables occurs. At the other hand, our theorem states that a variable must be marked with '+' in a mode if it is not of a type in OI. According to the definition of '+', there must be another atom which contains this variable, so we derive a contradiction.

4 Conclusions

We have shown that mode refinement in combination with both traditional subsumption and Object Identity subsumption has undesirable properties. While for traditional subsumption no proper refinement operator exists, Object Identity restricts the expressiveness of single clauses too much to obtain properness. We propose to reduce the disadvantages of OI by only considering search spaces that do not violate primary key constraints. In most situations, this is a very desirable restriction as it restricts the full clausal language to expressions that make sense from a human user point of view. For these more restricted languages, we have given an outline of a proof which convinces us that, using a weak subsumption operator, it is not necessary to force Object Identity to all variables in order to obtain a proper mode refinement operator; this allows single clauses to express more interesting patterns.

We have implemented primary keys, weak Object Identity and mode refinement in our multi-relational data mining algorithm FARMER [8]. In experiments with a graph database, these features allowed us to restrict the search space to clauses that represent graphs with single labels on the edges. This reduced the number of clauses that FARMER had to consider, and resulted in significant speed-ups.

As our restricted language can be as large as a full clausal language —in this case our weak OI subsumption becomes full OI subsumption— our setup is a generalization of Object Identity. Weak subsumption is exactly in the middle between traditional subsumption and OI subsumption.

References

[1] H. Blockeel and L. De Raedt. *Top-down Induction of Logical Decision Trees.* 1997.

[2] L. Dehaspe and H. Toivonen. Discovery of frequent Datalog patterns. In: *Data Mining and Knowledge Discovery 3*, no. 1, pages 7–36, 1999.

[3] F. Esposito, N. Fanizzi, S. Ferilli and G. Semeraro. A generalization model based on OI-implication for ideal theory refinement. In: *Fundamenta Informaticae*, 47, pages 15-33, 2001.

[4] F.A. Lisi and D. Malerba. Towards Object-Relational Data Mining. In: *Atti dell'Undicesimo Convegno Nazionale su Sistemi Evoluti per Basi di Dati*, pages 269–280, Rubbettino Editory, Italy, 2003.

[5] S. Muggleton. Inverse entailment and Progol. In: *New Generation Computing*, 13, pages 245–286. 1995.

[6] S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming.* LNAI 1228. Springer, 1997.

[7] S. Nijssen and J.N. Kok. Faster Association Rules for Multiple Relations. In: *IJCAI-01*, pages 891–896, 2001.

[8] S. Nijssen and J.N. Kok. Efficient Frequent Query Discovery in Farmer. In: *PKDD-2003*, 2003.

Feature Selection for Future Classes Proof of Concept

Wendy van Olmen Bart Naudts

University of Antwerp, Middelheimlaan 1, 2020 Antwerp, Belgium

Abstract

Two observations lie at the basis of this work. Firstly, in certain applications the feature selection for a classification task is somehow permanent (e.g., hard-coded in a physical device), and changing the classification requirements (in particular, adding extra classes) afterwards typically has a negative effect on classification performance. Secondly, when evaluating different feature selections for a given classification problem, it happens that no particular selection stands out head and shoulders above the others. When in these circumstances, we propose to use the margin of choice in the selection process to try to take future addition of one or a few classes into account. Making use of tree-based multinets as classifiers, a branch and bound search algorithm and a log likelihood separation measure to evaluate feature subsets, we show in this paper that this approach can be worthwile.

1 Motivation

Many recent advances in technology enable the generation of large amounts of data to be interpreted. Microarrays, for instance, allow a view on the activity of a cell by measuring the expression level of thousands of genes simultaneously [7]. Hyperspectral sensors can record images in up to 400 spectral channels at the same time, producing an enormous datacube [2]. Examples are abound.

The data produced by these technologies are often used for automatic classification. However, using all of this information to classify studied instances is not a good idea: not all of the data is equally relevant for the classification problem at hand, and the existence of dependencies between features may introduce redundancy among them. Taking into account all of the information available would make it hard for the classifier to discover the real distinguishing characteristics, causing it to be overly specified on the examples it was trained with, and hence reducing its generalizing power drastically. This suggests the benefit of a preprocessing step, selecting the appropriate features to base classification on [6]. Since selecting features comes at a cost, the objective is to select a minimal subset of features such that the classification will be equally, if not more accurate. This feature selection process may be quite complex.

As in many engineering problems, classification requirements may change over time. In particular, new classes are discovered and one would like the classifier to recognize them as well. This would generally involve selecting a new set of

appropriate features. However, apart from being inefficient, in practice it may not even be possible, because in certain applications the feature selection is somehow permanent: given the classification requirements, a feature selection is made, and this choice is hard-coded into a physical device accepting or gathering only inputs of that form, which is then used to perform high-throughput classification. The only remaining possibility is then to retrain the classifier based on the same set of features, which is bound to lead to unsatisfactory performance.

This gives rise to the central question of this paper: *Is it possible to foresee a few classifier extensions and anticipate to these changing requirements during feature selection?* When evaluating different feature sets, it often happens that not one particular selection stands out head and shoulders above the others (in particular, when there are insufficient training instances or when a lot of dependencies exist between features). We propose to use this margin of choice in the selection process to take the addition of one or a few future classes into account.

In this paper we show, using an artificial example, that candidate feature sets which are almost equally optimal with respect to the given task can really be distinguished from each other by looking at how they are expected to perform when one new class is added after the selection. In other words, we show that *being a bit less greedy at the time of selection may pay off later when a new class is added.* Essential here is of course the information about future classes, which we see probabilistically, as a distribution on a not necessarily finite set of classes.

We start by defining classification (Sect. 2) and feature selection (Sect. 3) before discussing the distribution on the future classes (Sect. 4). Section 5 contains the details of the experiment, and Sect. 6 concludes.

2 Classification

Classification is the problem of assigning a class label to instances or observations described by a set of features. In what follows, an instance will be represented as a vector of feature values $\vec{x}$.

We look at classification from a probabilistic point of view [4], and denote by $P(\vec{x}|c)$ the *class-conditional density function* defining class c, and by $P(c)$ the probability of occurrence of class c. The probability of misclassifying an instance $\vec{x}$ is then minimized by assigning it to the class with the largest posterior probability $P(c|\vec{x})$, with

$$P(c|\vec{x}) \sim P(\vec{x}|c)P(c)$$

according to Bayes' rule. In practice, both the underlying class definitions $P(\vec{x}|c)$ and class frequencies $P(c)$ are estimated from a set of preclassified instances D. Variations to this approach depend on the model used to represent $P_D(\vec{x}|c)$.

In this work, we use a *tree-based multinet* as a classifier [5]. This is an immediate generalization of the *tree augmented naive Bayes (TAN)* approach [5]. In TAN, the correlations between the features are modeled as a tree. The assumptions that there are only pairwise dependencies between variables and that the graph of these dependencies does not contain cycles, are made to render computation

of the classifier feasible [3]. Multinets allow different augmenting edges, different tree structures, for each class.

Note that, from the viewpoint of reducing the classification error rate, there are no bad features for a Bayesian classifier. This is because we cannot improve predictive accuracy of the theoretical classifier by eliminating a feature. However, there is usually insufficient data to reliably estimate all the probabilities and statistics that this approach requires. As a consequence, it is possible to improve performance by removing redundant and/or irrelevant features.

3 Feature Selection

For a comprehensive discussion on feature selection, we refer to [6]. We view feature selection as a *search problem* , with each state in the search space specifying a subset of the possible features.

In finding an appropriate subset, a number of criteria can be used for evaluation, among which *class separability.* Separability measures (also known as divergence, discrimination or distance measures) are constructed to measure the distance between the class-conditional density functions. Feature selection aims to separate the classes as much as possible, in order to minimize the number of misclassifications (especially when the data is subject to noise).

Given a sample $\{\vec{x}_1, \ldots, \vec{x}_N\}$ drawn from a reference class's distribution, an obvious choice for the distance to a second class c is the *average log likelihood* of this sample with respect to the second class's distribution, i.e.,

$$\frac{1}{N} \sum_{i=1}^{N} \log P(\vec{x}_i|c) \ .$$

In the experiment described in Sect. 5, we use this asymmetric distance measure. Note that by subtracting from this value the average log likelihood of the same sample with respect to the distribution is was itself drawn from, we obtain an approximation of the well-known Kullback–Leibler divergence [1].

4 Future Classes

It is obvious that taking into account future classes only makes sense when there is sufficient information that renders certain classes more likely to be added than others. Information about a future class may take the form of knowledge about specific values for certain features, (un)likely feature value combinations, interactions between features, the number of relevant features, the "size" of the class (is it a very narrow, or a broadly spread distribution?), etc. All of this information should be incorporated such that high probability "neighbourhoods" in the class distribution space arise.

In theory, we would use information to bias the uniform distribution on all class distributions (in our case, e.g., all tree-based Bayesian networks). In practice, however, it is not at all easy to describe non-trivial distributions on distributions. In

the particular case of the class distributions being fully factorized, as used by the naive Bayes classifier [5] , it is possible; the distribution may for example take the form of a probabilistic network whose variables control the parameters of each independent feature's distribution. As soon as graphical models come into play, however, no generally applicable methodologies exist to our knowledge. In the next section we will randomly generate a fixed number of class distributions. Each distribution is used as a representative for such a high probability "neighbourhood". We are aware that this is a serious approximation, but we leave it to further work to construct more realistic situations.

Once we have obtained a distribution over the future classes, however, it is at least in theory not too difficult to construct a new criterium that takes the addition of a future class into account. Formally, let $\mathcal{C}$ be a set of probability distributions over an observation space Ω based on values for n features. Consider performing feature selection for a set of (present) classes $\mathcal{B} \subset \mathcal{C}$, with a score function $\gamma(s|\mathcal{B})$ to evaluate a candidate feature subset $s \in \{0,1\}^n$ for the given classification task $\mathcal{B}$. Now, assume we have some prior knowledge about possible future classes $P(c)$, $c \in \mathcal{C}$. Our suggestion is to combine the score function $\gamma(s|\mathcal{B})$ with an extra selection criterium which computes the expected performance of the selection over the extended set of classes $\mathcal{B} \cup \{c\}$:

$$E_{\mathcal{C}}[\gamma(s|\mathcal{B} \cup \{\cdot\})] - \int \gamma(s|\mathcal{B} \cup \{c\}) \mathrm{d}_P c \ .$$

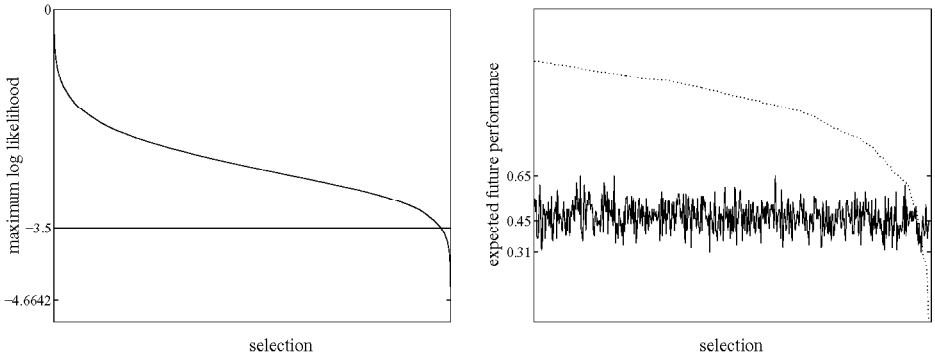
We will call this the *expected future performance* of a feature set. Higher order criteria, taking into account combinations of future classes, can be generalized immediately from this. (However, the more future classes expected, the more features that will end up being selected.)

5 Experiment

In this section, we will look at an example case of performing feature selection for classification. With this example, we will show that a significant improvement of the future performance is possible by using the proposed evaluation criterium.

Feature Selection Problem. We consider the artificial problem of selecting a subset out of 20 binary features to describe the instances to be classified. Assume the cardinality of the selected feature set has to be limited to a maximum of 5.

Classification Task. We use tree-based multinets to model the class-dependent densities over the feature values, as described in Sect. 2. In this work, being a proof of concept, we do not model the classes based on labeled training data, but directly generate the probability densities instead. (More details about how this is exactly done and properties of the resulting classes are given in the appendix.) We randomly generate 100 trees over the full set of features, of which 5 are picked to be the present classes of our classification task. The remaining trees represent the possible future classes and are assumed equally likely. Feature selection being a preprocessing step, the eventual classifier does not use these "fully defined" class distributions. Given a certain feature selection, we marginalize away the features that are not selected and consider this the learned class distributions instead.



(a) Maximum log likelihood for all 21700 selections of 5 features or less (out of 20); 517 values below the marked threshold

(b) Expected future performance for the 517 candidates (shifted max. log likelihood superimposed showing trade-off)

Figure 1: Results example case

Selection Criterium. We look at the weakest link for classification: the smallest distance between all pairs of (present) classes. Using the average log likelihood described in Sect. 3 as a measure of distance, we want the maximum log likelihood value to be below a threshold $\tau = -3.5$, i.e. we want the observations from a high-probability sample of one class to have an average probability of maximum $e^{\tau} = 0.03^*$ for any other (marginal) class distribution.

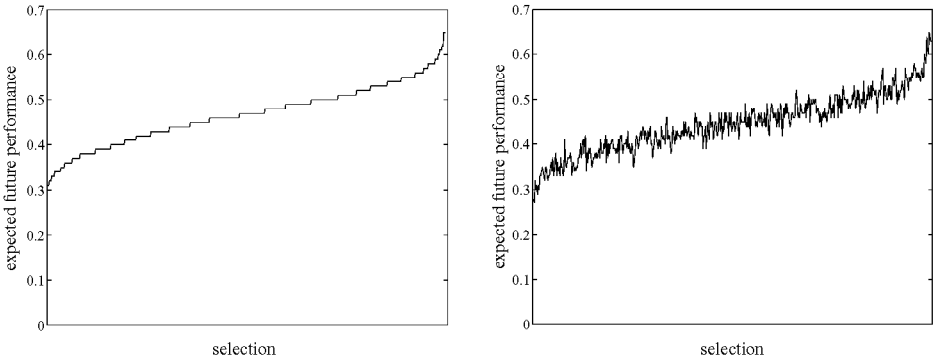
Expected Future Performance. In the context sketched above, the criterium suggested in Sect. 4 translates to: the percentage of possible future classes for which the average log likelihood would not exceed the threshold when added to the classification task as a sixth (present) class.

Results. This example being reasonable small, we calculated the maximum log likelihood for all possible feature subsets with cardinality smaller than or equal to five. The ordered values are shown in Fig. 1 (a), together with the set threshold: 517 out of 21700 selections suffice. For this margin of candidate feature selections, we plotted the expected future performance in Fig. 1 (b) (in the same order). With values ranging from 0.31 to 0.65, this plot shows that being a little less greedy as far as the original evaluation criterium is concerned, can improve the expected performance for future classes significantly: there are several near-optimal solutions that will remain usefull for 65 percent of the classifications tasks including one of the possible future classes, as opposed to the optimal solution for the present classification task which *fails* in 55 percent of all cases.

Implementation. A straightforward implementation of feature selection for future classes is to use the *branch and bound* algorithm† to search the space of

*Given a uniform distribution over the observation space based on 5 binary features, every observation has a probability of $\frac{1}{2^5} \approx 0.03$.

†Branch and bound is a variation of depth-first search that prunes branches extended from a node that evaluates worse than a bound value. Under the condition that the evaluation criterion is



(a) (Sorted) expected performance of the 517 candidate selections for the 95 originally generated future classes

(b) Expected future performance of the 517 candidate selections for a set of original class neighbours (in the same order)

Figure 2: Expected future performance for neighbouring sets of future classes

subsets, starting with all features selected and sequentially adding or deleting one feature, while considering a *multiple bound*: on one hand, a threshold is set for the maximum log likelihood between classes, and on the other hand, the best expected future performance found is kept up-to-date. After all, the probability of an instance (consequently, the average log likelihood and the maximum of the average log likelihood values) increases when the marginal distribution corresponding to an enclosed feature subset is considered, as the observation space in fact becomes smaller. Moreover, if the original criterium for evaluating feature subsets is monotonic, the extra criterium for considering future classes is monotonic as well.

Distribution on Future Classes. Since defining a distribution over all possible future classes is not a trivial problem, we suggested in Sect. 4 that taking into account the entire distribution might be approximated by considering only a finite number of classes representative for high probability neighbourhoods. To confirm this intuitition, we generated close neighbours for the 95 future classes by subjecting all randomly generated probabilities (Appendix) to Gaussian noise with a standard deviation of one percent , and repeated the experiment for this changed setting. The candidate solutions remaining the same, we compared how they are expected to perform for both sets of future class representatives. Figure 2 shows that most of the feature subsets score more or less the same on this extra criterium. Hence, how feature selections are expected to perform for a certain set of future classes, in general, appears to be representative for their behavior in the presence of future classes from the near neighbourhood. So, when high-probability neighbourhoods can be identified, this approximation might not be so bad .

monotone, i.e. nonincreasing or nondecreasing when considering enclosed subsets, this algorithm performs a complete search (no optimal subset is missed), since monotonicity ensures the pruned branches will not do any better.

6 Conclusion

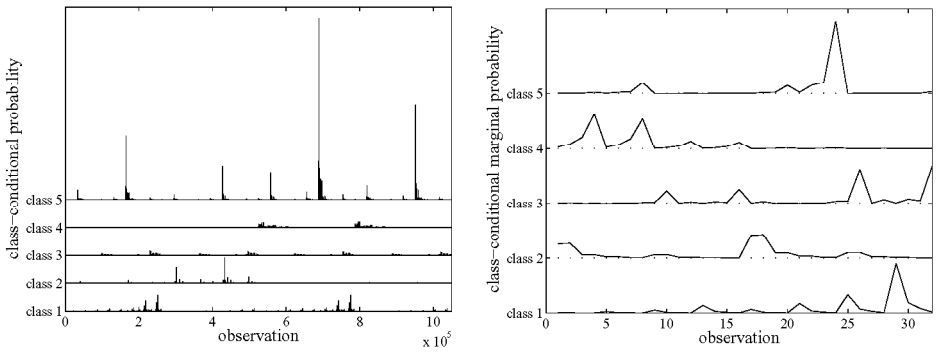
We motivated in this paper that in some cases there is an opportunity to obtain feature selections that can handle extensions of classification requirements. In Sect. 4 we proposed an extra selection criterium that takes into account the performance of feature subsets in the presence of possible future classes. On the basis of the small, though reasonably general example worked out in Sect. 5, we showed that the suggested criterium can really distinguish between candidate feature selections. Under those circumstances, we were able to improve the expected future performance significantly.

This paper being a proof of concept only, we have not investigated a number of important issues yet. Firstly, our distribution over the future classes possibly does not bear enough resemblance to real world cases. Since defining non-trivial distributions over graphical models is an open problem, we plan to investigate the impact of information about future classes on fully factorized class distributions first. Secondly, we need to quantify the trade-off between choosing the optimal selection and one with a better expected performance with respect to an additional class. Finally, we have planned to carry out a real world example in the area of bio-informatics (micro-arrays studies, centered around the selection of genes for toxic compound classification).

Appendix

The tree-based class distributions we used in Sect. 5 are generated by arbitrarily adding edges, under the condition that these edges do not introduce cycles. (We only add a random number of at most 10 edges, so, strictly speaking, we do not generate a connected tree, but a forest, i.e. a graph that consists of separated trees and independent nodes.) The conditional probabilities corresponding to the edges are set to cause strong correlations between the features involved: given a value of the parent feature, one arbitrary value of the dependent feature is assigned a probability of 0.9 and the other values are set to be equally (un)likely. Finally, the univariate probabilities of the root(s) and the independent features are filled in randomly, while the remaining probabilities are derived from these.

In general, the distributions generated this way do not have a very pronounced centre; instead, the probability mass is smoothened out over the entire observation space. To give an idea, we plotted the probability of all possible observations for the 5 present classes (Fig. 3 (a)). The problem with these smooth functions is that, given an observation, it is not clear which class it should be assigned to, because the probabilities are not very distinct. In the figure, however, the little peaks suggest combinations of feature values that are characteristic for the corresponding class. Feature selection can be seen as zooming in on these peaks: the distinctions between the classes are enlarged by considering the marginal distributions over the features that are not selected to represent a class. For example, in Fig. 3 (b) the marginal probabilities for the optimal subset according to the maximum log likelihood criterium of Sect. 5 are shown.



(a) Probability of all possible observations for the 5 present classes (plotted above each other for convenience). The observations are ordered in the classical (binary) way, from $0 = 00\ldots0$ to $1048579 = 11\ldots1$. All values lie between 0 and 0.072 and are represented on the same scale.

(b) Marginal probability of all possible observations for the feature subset with indices in $\{2, 3, 13, 16, 18\}$ (for convenience plotted above each other for the 5 present classes). The observations are ordered in the classical (binary) way, from $0 = 00000$ to $31 = 11111$. All values lie between 0 and 0.60 and are represented on the same scale.

Figure 3: Class-conditional probabilities

References

[1] H. Akaike. Information theory as an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki, editors, *Second International Symposium on Information Theory*, pages 267–281, Akademiai Kiado, Budapest, 1973.

[2] D. Casasent and X. W. Chen. Waveband selection for hyperspectral data: optimal feature selection. In *Proceedings of SPIE, Automatic Target Recognition*, Orlando, April 2003.

[3] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, May 1968.

[4] J. H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Machine Learning*, 29:103–130, 1996.

[5] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[6] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and data Mining*. Kluwer Academic Publishers, Boston Dordrecht London, 2000.

[7] E. P. Xing, M. I. Jordan, and R. M. Karp. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

Wind energy production forecasting

Floris Ouwendijk [a] Henk Koppelaar [a] Rutger ter Borg [b]

Thijs van den Berg [b]

[a] Delft University of Technology, PO box 5031, 2600 GA Delft, the Netherlands
[b] Nuon Energy Trade and Wholesale, Spaklerweg 20, Amsterdam, the Netherlands

Abstract

This paper presents a study to forecast the total production of some wind parks in The Netherlands. The predictions are based on several forecasts of the wind speed and wind direction. Two techniques are used: k-nearest neighbours and Lasso. These techniques are applied to three versions of the problem. The initial version scales the features. The second version modifies the wind directions to a more natural data format. The third version incorporates lags in the dataset. Each version improves the results somewhat, with the k-nearest neighbour technique having the lowest errors, yielding a large improvement over the current model.

1 Introduction to the problem

Nuon is the biggest wind energy producer in The Netherlands. Approximately 50% of the 600MW installed wind energy in The Netherlands is managed by Nuon. The produced energy is sold 24 hours in advance on the energy market. It is therefore important to know the amount of energy that will probably be produced, for differences between predicted and delivered amounts either cost money or result in a price that is sub optimal. The cost of power imbalance is around 6 Euro/MWh. This means that each MW of wind energy is worth 6 Euro less than that produced by perfectly predictable energy sources.

For a part of the total wind park, hourly data is available for the last two years. Predictions are based on predicted wind speeds and wind directions for different locations in the Netherlands. These weather predictions are also available 24 hours in advance. Because of the inherent error in these data, it will be impossible to create a perfect forecast.

This paper presents an attempt to improve the current predictions of the energy amount produced by the wind parks.

2 Previous work

There are not many articles concerning the application of machine learning techniques to the prediction of complete wind farms.

Li et al. [3] use neural networks for the prediction of the energy output for one turbine. Their results match the actual performance.
Denison et al. [2] predict the wind energy production on the longer term, based on historical data, by using a Bayesian multivariate adaptive regression spline model.
Nielsen and Nielsen [4] use statistical models to create production forecasts using meteorological data for wind farms. This led to the creation of the 'Wind Power Prediction Tool (WPPT)'.
Beyer et al. [1] use the numerical weather prediction model 'High Resolution Limited Area Model (HIRLAM)' for wind parks in Germany. Error rates of 12 to 20% are reported.

Unfortunately, because of the lack of articles regarding the same type of problem, it is hard to compare the results, or to use experience from previous results. We have been able to measure the error on the same dataset using the current model in use at Nuon. These results are described in the section on the current model.

3 Dataset description

The input features are the forecasted wind speed and wind direction of four locations in the Netherlands (DeKooy, Hoogeveen, Leeuwarden and Schiphol), as well as an average of the forecasted wind speed and wind direction at fifteen sites. These values are available 24 hours in advance from the national weather agency. The dataset consists of 18935 examples of data. This is the data for the period February 2001 to March 2003. Each example represents one hour of data in that year. Each example consists of ten input features and one output. The wind speed is measured in meters per second, discretized to 1 m/s intervals. The wind direction is measured in degrees. The locations of the forecasts and the locations of the wind parks are not the same. The output feature is the total energy output of a set of wind parks in the given hour.

Name	Output?	min	max	avg	st.dev
Direction NL	no	0.00	360.00	188.58	91.83
Direction DeKooy	no	0.00	360.00	186.81	95.41
Direction Hoogeveen	no	0.00	360.00	184.82	92.07
Direction Leeuwarden	no	0.00	360.00	185.79	94.49
Direction Schiphol	no	0.00	360.00	184.86	94.60
Speed NL	no	0.00	13.00	4.38	1.91
Speed DeKooy	no	0.00	15.00	5.73	2.42
Speed Hoogeveen	no	0.00	23.00	4.25	1.88
Speed Leeuwarden	no	0.00	21.00	4.63	2.01
Speed Schiphol	no	0.00	14.00	5.19	2.24
Production	yes	−72.00	84235.00	18366.91	19524.47

Table 1: Summary of the features in the dataset.

Correlations of the features show that the wind directions as well as speeds are highly correlated, which was expected, as the Netherlands is not a big country. The output is highly correlated with the wind speeds, and slightly with the wind directions.

Plots projecting each of the features against others, show a seasonal pattern where the wind speed or energy drops during July and August (figure 1). The wind speed is highest when the direction is around 230 degrees (figure 2 left). The plots for 'wind speed vs output' show that the wind speed is only an upper bound for the production (figure 2 right). For average wind speeds the production varies between zero and the upper bound. At high wind speeds the turbines have to be shutdown to prevent them from being damaged.

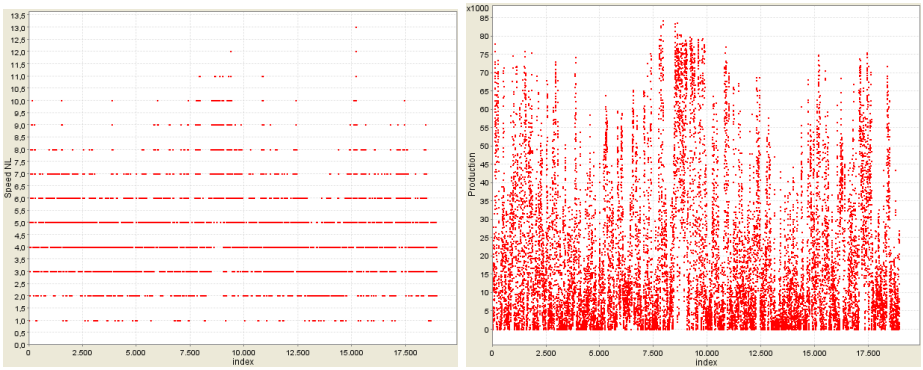


Figure 1: Distribution of wind speed and production over 2 years.

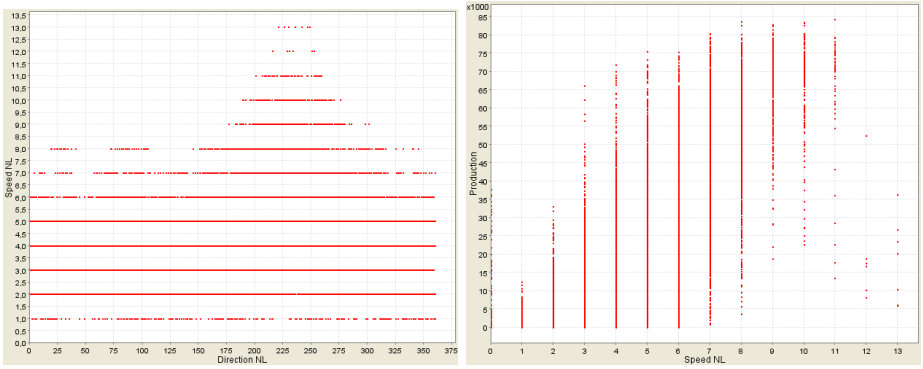


Figure 2: Left: wind direction vs wind speed, right: speed vs production

4 Our approach

This research was performed using the 'Workbench for Machine Learning Techniques' [5]. This is a tool that allows the user to easily create test scripts for testing

machine learning techniques and applying these techniques to problems such as this wind energy prediction problem. These scripts consist of modules placed in a stackable architecture.

Each of the modules in the script modifies the dataset. The modified dataset is passed on to subsequent modules, or used to apply a machine learning algorithm to a train and test set. The user-interface allows the parameters to the modules and learning techniques to be varied. The execution of the techniques is distributed over multiple computer systems. After the execution, different error measures are available. These measures can be set-off against the parameters using visualization modules. This allows us to quickly find optimal parameters.

Two machine learning techniques are used. The first is the k-nearest neighbour algorithm for regression problems. This technique was chosen because it is a very basic technique. The algorithm is initialised with a set of examples x_i and their corresponding output y_i. The distance to all examples is calculated for each new vector:

$$d(x_i, x_j) = \sqrt{\sum_{m=1}^{N} (x_{im} - x_{jm})^2}$$

The examples are ordered according to this distance, and the k examples with the least distance are selected. The predicted value for the new vector is then either the average of these points, or a weighed sum where examples with a smaller distance contribute more to the predicted output.

The second technique is Lasso [6]. This technique was chosen for it's good results on other problems, and to see how it would perform on this problem. Lasso is based on relevance vector machines, which produce sparse models using kernel functions. It has the advantage that the error distributions for train set and test set are comparable. Predictions are calculated by

$$f(x) = \alpha_0 + \sum_{i=1}^{n} \alpha_i K(x_i, x)$$

in which α is a vector weights for each of the examples, and K is the kernel function. To obtain α, the problem

$$\begin{aligned} \text{minimise} \quad & ||y - 1\alpha_0 - \mathbf{K}\alpha||_2^2 \\ \text{subject to} \quad & ||\alpha||_1 \leq \text{kappa} \end{aligned}$$

is solved. Kappa controls the complexity of the solution. Two kernel functions were used; the radial basis function (RBF) kernel and the polynomial kernel.

$$\begin{aligned} \text{Polynomial} \quad & K(u, v) = (\text{gamma}\ u \cdot v + \text{coef})^d \\ \text{RBF} \quad & K(u, v) = e^{-(u-v)^2/\sigma^2} \end{aligned}$$

Three approaches are taken to improve the predictions that are currently obtained (section 4.1). The first approach is to use scaled data. The second modifies the wind directions in the dataset to a more natural system. The third approach extends the dataset even further by adding lags of the input features to the dataset.

The script used in the workbench consists of several modules. The first module in the script is the 'normalize' module. This module scales all inputs and the output to mean 0 and standard deviation 1. This choice stems from the practice that making the influence of the features comparable is a good first guess. After this scaling all values lie between -1 and 3.

The second module is a 'random subset' module. This module picks a random set of examples from the dataset. The size of this set was set to 30%. The reason to do so is that the dataset contains a large number of examples, of which many are similar. This slows down the training while the results are assumed to be similar when compared to training on the complete dataset. During the tests, this assumption will be tested.

The third module is a cross-validation module that creates four folds. This allows us to see if there are large variations in the results. Four folds were chosen as a trade-off between the number of tests and the inclusion of similar examples in the train and test sets. With fewer folds, the chance to have similar examples in both the train set and the test set decreases.

The fourth and final module is the machine learning technique. For the first attempt, this is initially the k-nearest neighbour module, where k can be varied, and has two weighing modes. Thereafter the Lasso module is used, which allows kappa to be varied, as well as choosing a kernel function and its parameters. In search of a better model, the parameters of the previous test are optimized each time after the dataset is modified. This optimization consists of varying multiple parameters together (trying all combinations of values in specified intervals), first at a coarse scale, and then on a finer scale. This typically results to five to ten values being tested per parameter per try. And multiple tries (on a finer and finer scale) to get to an optimum.

4.1 Current model

The current model in use at Nuon is a third order polynomial. This polynomial is fit on the complete dataset to obtain the least mean squared error solution.

Using this model, predictions for all examples in the dataset were obtained. This results in a mean absolute error of 0.44 and a mean squared error of 0.36 between the actual and predicted values.

4.2 Basic approach

The first attempt to improve the current model is by using k-nearest neighbours. The workbench was given a script that modified the dataset (normalize all features and output), and to search for the best settings. These were determined to be at $k = 6$ with the nearest neighbours being weighted by their distance to the given example and then summed. The mean squared error for these settings was 0.04 and on the test set 0.29. With these settings fixed, the test was repeated with the complete dataset. The error dropped to the score given for test 1 in table 2.

The next tests are performed using Lasso. They are based on the same script in the workbench, with the k-nearest neighbour module being replaced by the Lasso

module. Two kernels were tried: the RBF and the polynomial kernel.

The RBF kernel was tried for different values of kappa and sigma. In the Lasso algorithm, kappa is the upper bound for the L1 norm of the active set. For different values for kappa, there were always some values of sigma that performed best. The differences between these pairs however are negligible. For both train and test set, the MSE varied around 0.31.

The polynomial kernel was tried for different degrees, and different values for gamma and the coefficient, in addition to kappa. The polynomial of the fourth degree was the most effective. For parameters kappa=0.7, gamma=0.3 and coefficient=3.0 the best results were obtained (table 2, test 2).

When the amount of examples that is allowed in the dataset is increased, these values do not change.

4.3 Modified wind directions

The wind directions varied over 0 to 360 (before scaling them around zero). The jump from 360 to 0 is quite significant, whereas in reality there is no distinction. To clear this problem, the directions were all mapped onto a circle. That way, the values become better comparable. The original wind directions are then removed from the dataset, and sine and cosine values of the mappings are added.

To measure the results, the polynomial kernel was used, again with degree 4. The initial results (with all settings kept the same) improved over the previous test.

When the number of examples that was used was increased to 70% the numbers did not change. By varying the values for kappa, gamma and the coefficient, it was found that the parameters were still optimal. Results are in table 2, for test 3.

4.4 Adding lags

Because of the discretized wind speed values, we assume that better predictions are obtained by incorporating lags of previous wind speeds. The script was modified so that after the mapping of directions onto circles all input features were lagged for periods 1, 2 and 5 time steps. For lag=5, this resulted in a dataset consisting of 90 input features (for each of the five locations, the wind speed and the sine and cosine of the direction are available, and these values are repeated for t-1, t-2, t-3, t-4 and t-5).

The tests were performed with the same settings as for the previous test. The results are summarized in table 2, as tests 4, 5 and 6.

Because of the good initial results with k-nearest neighbours and the varying results with Lasso, an additional test was performed with the kNN module, with lag=5 on 30% of the examples. This resulted in the measures in table 2, test 7.

As results for kNN improved as more examples were used for training, the test is repeated with all examples, and a lag of one time step. These results are summarized in table 2, test 8.

Test	Train set			Test set		
	Avg	Min	Max	Avg	Min	Max
Test 1 MSE	0.03	0.03	0.03	0.23	0.22	0.24
Test 1 MAE	0.11	0.11	0.11	0.32	0.31	0.32
Test 2 MSE	0.29	0.29	0.29	0.30	0.29	0.31
Test 2 MAE	0.39	0.38	0.39	0.39	0.39	0.40
Test 3 MSE	0.26	0.25	0.28	0.27	0.27	0.27
Test 3 MAE	0.37	0.36	0.38	0.37	0.37	0.38
Test 4 MSE	0.25	0.24	0.26	0.29	0.26	0.31
Test 4 MAE	0.36	0.36	0.37	0.39	0.37	0.39
Test 5 MSE	0.26	0.26	0.27	0.31	0.29	0.32
Test 5 MAE	0.37	0.36	0.37	0.39	0.38	0.40
Test 6 MSE	0.25	0.24	0.26	0.30	0.28	0.32
Test 6 MAE	0.36	0.36	0.37	0.39	0.38	0.40
Test 7 MSE	0.05	0.04	0.05	0.24	0.23	0.24
Test 7 MAE	0.14	0.14	0.14	0.34	0.33	0.34
Test 8 MSE	0.03	0.03	0.04	0.20	0.19	0.20
Test 8 MAE	0.12	0.12	0.12	0.29	0.29	0.30

Test 1: kNN, scaled dataset
Test 2: Lasso, polynomial kernel, scaled dataset
Test 3: Lasso, polynomial kernel, modified wind directions
Test 4: Lasso, polynomial kernel, modified wind directions, lag of one time step
Test 5: Lasso, polynomial kernel, modified wind directions, lag of two time steps
Test 6: Lasso, polynomial kernel, modified wind directions, lag of five time steps
Test 7: kNN, modified wind directions, lag of five time steps
Test 8: kNN, modified wind directions, lag of one time step, complete dataset

Table 2: Summary of test results

5 Conclusion

Two techniques were applied to the Nuon wind speed dataset. These techniques were k-nearest neighbours and Lasso. Successive improvements were made to the dataset to obtain better results.

The k-nearest neighbour technique was able to obtain very good results. This is probably due to the huge amount of examples. Good examples are always near and the pattern is obviously stable.

Lasso is a sophisticated technique. From the comparison with the kNN technique, it seems that modelling the results of the predictions on the energy output is hard. Further research is needed to see if other kernels are able to improve the results.

Conversion of the wind directions to a more natural format improved the predictions. Results for adding lags vary. Apparently the addition adds less information to the data than the increase in complexity is able to justify. Adding a small amount of lag improves the results (minimum average MSE and MAE drop

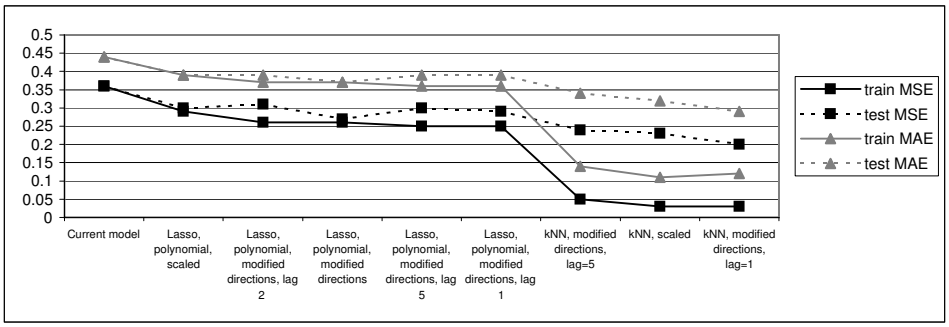


Figure 3: Results of the tests

slightly), but the average errors on the test set increase.

The use of Lasso with a fourth degree polynomial kernel results in average errors of 9-10%. By using k-nearest neighbours with distance based weighing of the 6 nearest neighbours the average error is expected to be in the range 3-8%. When the best average results on a test set are compared to the results of the currently employed method, a 34% improvement is achieved. With today's wind energy volumes this already represents a significant amount of money.

References

[1] H.G. Beyer, D. Heinemann, H. Mellinghoff, K. Monnich, and H. Waldl. Forecast of regional power output of wind turbines, 1999.

[2] D.G.T. Denison, P. Dellaportas, and B.K. Mallick. Wind speed prediction in a complex terrain, 2001.

[3] S. Li, D.C. Wunsch, E.A. O'Hair, and M.G. Giesselmann. Using neural networks to estimate wind turbine power generation energy conversion. *IEEE Transaction on energy conversion*, 16(3):276–282, September 2001.

[4] H. A. Nielsen and T. S. Nielsen. Using meteorological forecasts in on-line predictions of wind power. chapter 5: Estimation methods, 1999. ISBN: 87-90707-18-4.

[5] F.A. Ouwendijk. Workbench for machine learning techniques. Master's thesis, Delft University of Technology, june 2003.

[6] V. Roth. Sparse kernel regressors. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Proceedings of the International Conference Vienna (ICANN'01)*, volume 2130 of *Lecture Notes in Computer Science*, pages 339–346. Springer-Verlag, 2001.

Generating Artificial Data for Monotone Classification and Regression Problems

Rob Potharst

Erasmus University, P.O.Box 1738, 3000 DR Rotterdam

Abstract

Experience tells us that it is not trivial to generate artificial data for monotone learning problems. Two algorithms are presented in this paper for such learning problems. The first one can be used to generate random monotone data sets without an underlying model, and the second can be used to generate monotone decision tree models. If needed, noise can be added to the generated data. The second algorithm makes use of the first one. Both algorithms are illustrated with an example.

1 Introduction

When we develop or revise a new or existing method for solving a learning problem, we will be sure to test this method extensively, usually not only on real life data, but also on artificial data. One reason for the inclusion of artificial data in our experiments is that the parameters of the models underlying the data can be much better controlled for artificial data when compared with real data. For instance, if we want to test the robustness of our newly devised learning method against varying degrees of noise in the data, it is generally much easier to generate artificial data than to search for real datasets with varying degrees of noise. In many cases, artificial datasets can be constructed without much effort using simulation techniques [8], for instance by first generating random values for the parameters of our model, next generating random input values, calculating the corresponding outputs and adding random noise to the output values. However, artificial data generation is not always that easy. An example is the construction of datasets for concept learning problems [5]. The construction of monotone datasets is another example. The monotonicity constraint on a model or dataset has recently been studied by a number of authors [1, 7, 2, 4] and will be explained in Section 2. Application areas of monotone classification and regression include pricing and various selection problems [1, 7]. While working on algorithms for monotone classification and regression problems, we found that it is not a trivial task to come up with a monotone dataset, or a monotone classification or regression model. This has to do with the great number of constraints on the data values that is implied by the monotonicity constraint. In [6], an algorithm for constructing random monotone datasets is proposed, which has never been published in a regular medium. This algorithm is outlined in Section 2 of this paper. In Section 3 we present a new method to construct monotone classification or regression tree models, using the

method of Section 2 and a special order relation on the leaves of a tree introduced in [3]. This method is illustrated with an example in Section 4. The last section contains the summary of this paper.

2 Generating random monotone datasets

In this section we will propose a technique that produces monotone datasets with prescribed parameters. This technique could, for instance, generate a monotone data set of n elements from a given instance space $\mathcal{X}$ and a set of target values $\mathcal{Y}$, such that the frequency of the occurrence of these target values follows a prescribed pattern. (If $\mathcal{Y}$ is a finite set of classes, it is a classification problem, if it is a set of numbers, it is a regression problem.) This data set will be random, in the sense that it is a random sample from the space of all possible datasets satisfying the prescribed conditions. However, it is not guaranteed that all possible samples have equal chances of being drawn. In order to describe the technique we will first introduce a mapping between datasets and certain graphs, that will facilitate the description.

Let $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_p$ be an instance space with p attributes, $\mathcal{Y}$ a set of target values and let $\mathcal{D}$ be a set of examples $(\mathbf{x}, y)$ from $\mathcal{X} \times \mathcal{Y}$. A dataset is called *monotone* if for all pairs $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$ from $\mathcal{D}$ we have

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow y \leq y'. \tag{1}$$

With $\mathcal{D}$ we will associate a labeled directed graph $\mathcal{G}(\mathcal{D})$ as follows: $\mathcal{D}_x$, the set of $\mathbf{x}$-values from dataset $\mathcal{D}$, will be the set of vertices of the directed graph, and let $[\mathbf{x}, \mathbf{x}']$ be an arc[1] of the graph iff $\mathbf{x} > \mathbf{x}'$. Let further each vertex $\mathbf{x}$ of the graph be labeled with the target value y of data example $(\mathbf{x}, y)$. For example, the 3-attribute dataset $\mathcal{D}$:

001	0
002	1
112	2
202	2
212	3

will have as its associated graph $\mathcal{G}(\mathcal{D})$ the graph of Figure 1.

We will call a path in a graph $\mathcal{G}(\mathcal{D})$ *non-increasing* if the labels of the successive vertices on the path form a non-increasing sequence. For instance, the path $[212, 112, 002, 001]$ is non-increasing since the labelings form the non-increasing sequence $[3, 2, 1, 0]$. Even if we would relabel 002 to 2, the path would stay non-increasing; however if we would relabel 001 to 2 the path would loose this property. The following lemma connects the monotonicity of a data set with the non-increasing nature of the paths in its associated graph.

[1] An arc is an edge together with a direction.

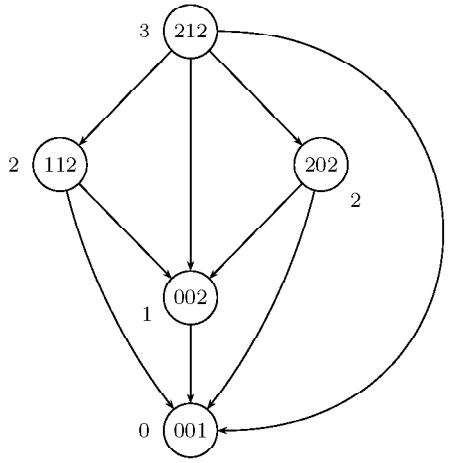


Figure 1: The Graph Associated with a Dataset

Lemma 1 *Let $\mathcal{D}$ be a data set on $\mathcal{X} \times \mathcal{Y}$ and let $\mathcal{G}(\mathcal{D})$ be its associated labeled graph. Then we have*

$$\mathcal{D} \text{ is monotone} \iff \text{all paths in } \mathcal{G}(\mathcal{D}) \text{ are non-increasing.}$$

Proof: First we prove the $\Rightarrow$ part. In that case we suppose that (1) holds for all $(\mathbf{x}, y) \in \mathcal{D}$. Now, suppose we have a path $[\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n]$ in $\mathcal{G}(\mathcal{D})$, then $[\mathbf{x}_0, \mathbf{x}_1]$,$[\mathbf{x}_1, \mathbf{x}_2]$, etc. must be arcs of the graph, so from the definition of the associated graph it follows that $\mathbf{x}_0 > \mathbf{x}_1 > \ldots > \mathbf{x}_n$. Now, from (1) we can conclude that $y_0 \geq y_1 \geq \ldots \geq y_n$). Next we prove the $\Leftarrow$ part. So we suppose that all paths in $\mathcal{G}(\mathcal{D})$ are non-increasing. We must now prove (1) for all pairs $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$ from $\mathcal{D}$. So let $(\mathbf{x}, y)$ an $(\mathbf{x}', y')$ be data examples from $\mathcal{D}$ with $\mathbf{x} \leq \mathbf{x}'$. If $\mathbf{x} \leq \mathbf{x}'$ then $y = y'$, so (1) is trivial. If $\mathbf{x} < \mathbf{x}'$ then $[\mathbf{x}', \mathbf{x}]$ is an arc of $G(\mathcal{D})$ so $[\mathbf{x}', \mathbf{x}]$ is also a path. Since all paths are non-increasing it follows that $y' \geq y$.

We proceed with an informal description of the algorithm for producing monotone datasets. Suppose, the required datasets must have N examples. We start with selecting N different vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ randomly from $\mathcal{X}$. Next, we select N not necessarily different target labels $y_1, \ldots, y_N$ from $\mathcal{Y}$. Now, each of the target labels $y_1, \ldots, y_N$ must be assigned to exactly one of the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ to obtain the required data set. This will be done as follows:

- construct the graph associated with $\mathbf{x}_1, \ldots, \mathbf{x}_N$
- sort the sequence $y_1, \ldots, y_N$ such that $y_1 \leq y_2 \leq \ldots \leq y_N$
- now, interpret the graph as a flow network: each of the target labels $y_1, \ldots, y_N$ is allowed to travel through the network, following a random path; when the label comes to the end of its path, the target label is sticked to the last vertex on the path; this vertex is subsequently removed from the graph, and the next label is allowed to travel through the new graph. This is done until each vertex of the original graph has a target label sticked to it.

For instance, if the randomly drawn vectors are 001, 002, 112, 202, and 212 and the target labels are 0, 1, 2, 2, 3 we end up with the above graph $\mathcal{G}$, so the above data set $\mathcal{D}$ results. However, if with the same set of vectors we want to associate the target labels 0, 1, 1, 2, 3, one of the following datasets will result:

001	0
002	1
112	1
202	2
212	3

001	0
002	1
112	2
202	1
212	3

each with probability $\frac{1}{2}$.

We will end with a formal description of the algorithm for producing monotone datasets:

1. Draw $\mathbf{x}_1, \ldots, \mathbf{x}_N$ randomly, without replacement, from $\mathcal{X}$.
2. Select $y_1, \ldots, y_N$ from $\mathcal{Y}$, see Remark 1 below.
3. Order $y_1, \ldots, y_N$ such that $y_1 \leq y_2 \leq \ldots \leq y_N$.
4. Define the $(N+1) \times N$ matrix $M = (m_{ij})$ as follows: for $1 \leq i, j \leq N$

$$m_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_j < \mathbf{x}_i \\ 0 & \text{otherwise} \end{cases}$$

and for $1 \leq j \leq N$:

$$m_{0j} = \begin{cases} 1 & \text{if } \sum_{i=1}^{N} m_{ij} = 0 \\ 0 & \text{otherwise} \end{cases}$$

5. Perform the following algorithm on M:

```
for j := 1 to N do
    begin
        k := FindZinc(M);
        label(k) := y_j;
        for i := 0 to N do m_{i,k} := 0
    end
```

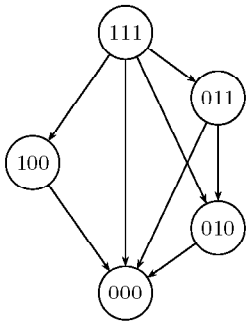
where the function FindZinc, which returns a number between 1 and N, is defined as follows:

```
FindZinc(var M):
    p := 0;
    while PathsFrom(p) is non-empty do
        p := random element from PathsFrom(p);
    return p
```

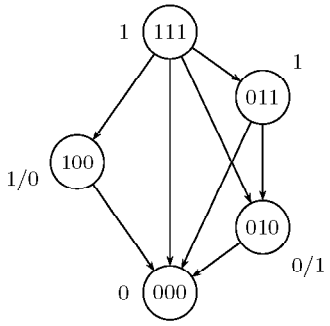
where the set PathsFrom(p) is defined as PathsFrom(p) :$-$ $\{i \in \{1, \ldots, N\} : m_{pi} > 0\}$.

Remark 1 Note, that in step 2 of this algorithm, we can fill in our own selection mechanism. For instance, we could select the target labels according to a random mechanism or take some arbitrary set of labels at will.

Remark 2 It can be shown that for a given set of vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and an ordered list of target labels $[y_1, \ldots, y_N]$ the above algorithm can generate each possible monotone data set $\mathcal{D}$ with $D_x = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $D_y = [y_1, \ldots, y_N]$. Sometimes, it also generates all the possible monotone datasets with equal probability, as is the case in the above example. However, this is not necessarily the case, as can be seen from the following example: If the set of vectors is 000, 100, 010, 011, 111 the associated graph has the following shape



Now, if the target label list is $[0, 0, 1, 1, 1]$ we can generate two different monotone datasets, viz.



Here the notation 1/0 means class 1 for the first data set and class 0 for the second data set. Note that the first data set is generated with probability $\frac{2}{3}$ and the second one with probability $\frac{1}{3}$. Thus, the datasets generated by our algorithm, although guaranteed to be monotone, might contain a slight bias in the sense that some monotone datasets have a higher probability of being selected than others. This point needs further investigation.

3 Generating structured monotone datasets

In many cases, we want our artificial datasets to be structured or model-based. The reason for this is the following: usually, the algorithm we want to test on the artificial dataset is supposed to discover an underlying pattern or structure in the data; for instance, in classification or regression problems, a relation between the attributes and the target values is supposed to be discovered. Thus, our artificial dataset should be based on such an underlying relation or model: it is hard to discover something, when there is nothing. In case we deal with a monotone problem, the underlying model should be monotone. If our model is to be tree-based (as will be the models in this paper), how are we going to label the leaves in such a fashion that the resulting tree will be monotone? In this section we will show that to label the leaves, we can essentially use the algorithm of the preceding section, applied to the leaves of the tree, provided these leaves are ordered according to the special ordering, introduced in [3].

Specifically, if $\mathcal{T}$ is a decision tree on instance space $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_p$ with instances $\mathbf{x} = (x_1, \ldots, x_p)$ and the test in each node of the tree has the form $X_i \leq c$ for some $c \in \mathcal{X}_i, 1 \leq i \leq p$, then the associated subset $T \subset \mathcal{X}$ of each node or leaf has the form $T = \{\mathbf{x} \in \mathcal{X} : \mathbf{a} < \mathbf{x} \leq \mathbf{b}\} = (\mathbf{a}, \mathbf{b}]$ for some $\mathbf{a}, \mathbf{b} \in \overline{\mathcal{X}}$ with $\mathbf{a} < \mathbf{b}$ and $\overline{\mathcal{X}} = \mathcal{X}$ extended with infinity elements. We will call $\mathbf{a} = \min(T)$ the minimal element or left corner of node T and $\mathbf{b} = \max(T)$ the maximal element or right corner. The set of leaves $\mathcal{L}$ of the tree form a partition of $\mathcal{X}$ and are labeled with values from $\mathcal{Y}$. Thus, the tree defines an associated labeling rule $f : \mathcal{X} \to \mathcal{Y}$. The tree will be called *monotone* if for its associated labeling rule $f(\mathbf{x})$ we have

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow f(\mathbf{x}) \leq f(\mathbf{x}'). \tag{2}$$

The special ordering we need for nodes and leaves is the following: if T and T' are nodes or leaves, then we define

$$T \leq T' \iff \min(T) \leq \max(T'). \tag{3}$$

In [3] it is shown that this ordering is reflexive and anti-symmetric, but not transitive. Thus, it does not induce a partial ordering on the set of leaves. However, a partial ordering is not necessary for the following theorem to hold.

Lemma 2 *Let $\mathcal{T}$ be a classification or regression tree on $\mathcal{X}$ with leaves $\mathcal{L}$ and let f be its associated labeling rule. For all pairs $T, T' \in \mathcal{L}$ we have*

$$T \leq T' \Rightarrow f(T) \leq f(T'), \tag{4}$$

if and only if $\mathcal{T}$ is monotone.

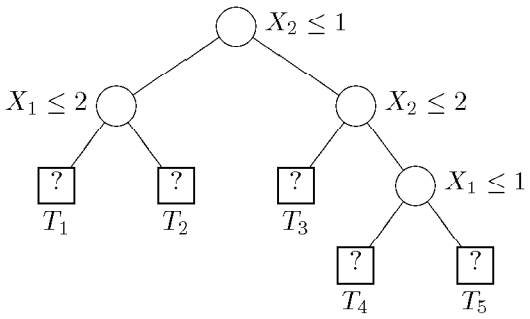
Proof: We first prove the 'only if' part. It is clear that with $f(T)$ we mean the labeling assigned to leaf T by tree $\mathcal{T}$. To prove that $\mathcal{T}$ is monotone we should prove (2) for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. If $\mathbf{x}$ and $\mathbf{x}'$ belong to the same leaf, (2) is trivial. So, let us suppose that they belong to different leaves, with T the leaf that contains $\mathbf{x}$ and T' the leaf that contains $\mathbf{x}'$. Since by hypothesis $\mathbf{x} \leq \mathbf{x}'$, we have $\min(T) \leq$

$\mathbf{x} \leq \mathbf{x}' \leq \max(T')$, so according to our definition (3) we have $T \leq T'$. Thus, by (4) we have $f(T) \leq f(T')$, which proves (2). The reverse part follows the same lines.

We can use Lemma 2 to assign labels to an unlabeled decision tree $\mathcal{T}$ with leaves $\mathcal{L}$ such that the resulting decision tree will be monotone, as follows. First, generate as many labels $y \in \mathcal{Y}$ as there are leaves in the tree. This can be done according to any distribution. Next, sort these labels in ascending order: $y_1 \leq y_2 \leq \ldots \leq y_k$. Next, we build the associated graph $\mathcal{G}(\mathcal{L})$ of the leaves just like we did we the instances in Section 2. Thus, the vertices of the graph are now leaves instead of instances. We now run the algorithm of Section 2 on this graph together with the labels $y_1 \leq y_2 \leq \ldots \leq y_k$. The proof that the resulting labeled tree will be monotone follows directly from Lemma 2.

4 Example

In this section we will show how we can use the method of Section 3 to label an unlabeled decision tree monotonically. Suppose we have the following unlabeled tree:

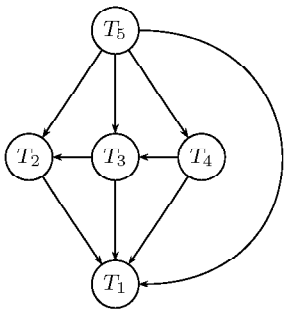


The corner elements $\min(T_i)$ and $\max(T_i)$ of the leaves of this tree are easily checked to be as follows:

leaf	$\min(T_i)$	$\max(T_i)$
T_1	$(-\infty, -\infty)$	$(2, 1)$
T_2	$(2, -\infty)$	$(\infty, 1)$
T_3	$(-\infty, 1)$	$(\infty, 2)$
T_4	$(-\infty, 2)$	$(1, \infty)$
T_5	$(1, 2)$	(∞, ∞)

By inspection of this table we can find all the arrows in the associated graph for these leaves, which is shown at the top of the next page.

Now, if we generate the following set of labels $\{0, 1, 1, 1, 2\}$, the algorithm of Section 2 returns the following label assignment: $f(T_1) = 0, f(T_2) = f(T_3) = f(T_4) = 1, f(T_5) = 2$ which gives indeed a monotone decision tree as can be easily checked.



By the way, we also see here an example of the non-transitivity of the special ordering of the leaves: we have $T_2 \leq T_3$ and $T_3 \leq T_4$, but $T_2 \not\leq T_4$ since $(2, -\infty) \not< (1, \infty)$.

5 Conclusion

The algorithms of this paper are an important aid to help us with the difficult task of generating artificial data for monotone models. The first algorithm is guaranteed to give a monotone data set, the second algorithm yields a monotone decision tree based on any unlabeled tree. This monotone tree can subsequently be used to generate structured monotone data; if needed, random noise can be superimposed. As a side effect, in this paper interesting characterizations of monotone datasets and monotone decision trees are proved.

References

[1] A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19(1):29–43, 1995.

[2] J.C. Bioch and V. Popova. Monotone decision trees and noisy data. In *ERIM Report Series Research in Management, ERS-2002-53-LIS*, 2002.

[3] J.C. Bioch and V. Popova. Induction of ordinal decision trees: an MCDA approach. In *ERIM Report Series Research in Management, ERS 2003 008 LIS*, 2003.

[4] A.J. Feelders and M.A. Pardoel. Pruning for monotone classification trees. In *Proceedings of the 5th International Symposium on Intelligent Data Analysis*, 2003. To appear.

[5] T.J.C. Hunniford and R.J. Hickey. A simulated annealing technique for generating artificial data to assess concept learning algorithms. *Intelligent Data Analysis*, 3(3):177–189, 1999.

[6] R. Potharst. *Classification using decision trees and neural nets.* PhD-thesis, Erasmus University, Rotterdam, 1999.

[7] R. Potharst and A.J. Feelders. Classification trees for problems with monotonicity constraints. *SIGKDD Explorations*, 4(1):1–10, 2002.

[8] Sheldon M. Ross. *Simulation, 3rd edition.* Academic Press, New York, 2001.

Multi-Agent Diagnosis with semantically distributed knowledge

Nico Roos [a] Annette ten Teije [b] Cees Witteveen [c]

[a] Universiteit Maastricht, IKAT, P.O.Box 616, 6200 MD Maastricht.
[b] Free University, Faculty of Sciences, De Boelelaan 1081A, 1081 HV Amsterdam.
[c] Delft University of Technology, ITS, PO Box 5031, 2600 GA Delft.

Abstract

We consider the problem of finding a commonly agreed upon diagnosis for errors observed in a system monitored by a number of different expert agents, each having its own specialized view on the system. That is, the expert agents have to agree on one or more diagnoses based on their specialized views of the system. Reaching an agreement is complicated by the two factors: (*i*) different specialisms need not distinguish the same fault modes of a component and (*ii*) knowledge of different specialisms need not be correct in some cases. This paper analyzes these problems and presents protocols that enable the agents to deal with these issues.

1 Introduction

A traditional diagnostic tool can be viewed as a single *diagnostic agent* having a model of the whole system to be diagnosed. In some applications, however, such a single agent approach is infeasible or at least undesirable. For example, the integration of knowledge into one model of the system is infeasible if the system is too large, too dynamic or distributed over different legal entities. Integration is undesirable if it concerns the combination of knowledge from different fields of expertise. In this latter case, where knowledge is called to be *semantically distributed*[1] [4], it would be better to introduce specialized agents communicating about anomalies detected.

The introduction of specialized (expert) agents immediately raises the problem how to reach an agreement on the cause of observed anomalies. As was pointed out in [7, 8], assuming a fixed maximum number of broken components, there exists a polynomial time protocol for reaching an agreement between the agents in case of a semantic knowledge distribution. This protocol is rather straight forward. A more difficult situation arises if the knowledge of some agents is *incomplete* in the sense that the agents have no behavioral knowledge about some fault modes, or if the knowledge of some agents is *incorrect* in the sense that the agents have incompatible knowledge about the behaviors of components. In this paper, we will address both issues.

This paper is organized as follows. Section 2 specifies the diagnostic setting, which is extended to multi-agent diagnosis in section 3. Section 4 introduces protocols for multi-agent diagnosis and section 5 concludes the paper.

[1] Besides a semantic knowledge distributed, we also distinguish a *spatial knowledge distribution*: knowledge of system behavior is distributed over the agents according to the spatial distribution of the system's components. The latter has been discussed in [9].

2 The diagnostic setting

A system to be diagnosed is a tuple $S = (C, M, Id, Sd, Ctx, Obs)$ where C is a set of components, $M = \{M_c \mid c \in C\}$ is a specification of possible fault modes per component, Id is a set of identifiers p of connection points between components, Sd is the system description, Ctx is a specification of input values of the system that are determined outside the system by the environment and Obs is a set of observed values of the system. A component in C has a normal mode $nor \in M_c$, one general fault mode $ab \in M_c$ and possibly several specific fault modes. We assume that all components have *in-* and *outputs*.[2]

The system description $Sd = Str \cup Beh$ consists of a structural description Str and a behavioral description Beh of the components. The structural description Str consists of instances of the form $p = in(x, c)$ or $p = out(x, c)$ where x is an in- or an output identification of a component c and $p \in Id$ is a connection point identifier[3]. Of course, a connection point $p \in Id$ is connected to at most one output of some component; i.e. if $p = out(x, c)$ and $p = out(y, c')$, then $x = y$ and $c = c'$. A connection point has a value, which is determined by the output of a component or a system input. The function $value(p)$ denotes the value of the connection point.

The set $Beh = \bigcup_{c \in C} Beh_c$ specifies a behavior for each component $c \in C$. The behavior description Beh_c of a component describes the component's behavior for each (fault) mode in M_c, possibly with the exception of $ab \in M_c$; i.e. $mode(c, ab) \rightarrow \top$. In this specification, the predicate $mode(c, m)$ is used to denote the mode $m \in M_c$ of a component c. For each instance $mode(c, m)$, Beh_c specifies a behavioral description of the form: $mode(c, m) \rightarrow \Phi$ where $m \in M_c$.[4] The expression Φ describes the component's behaviour given its mode $m \in M_c$.

The set Ctx describes the values of system inputs $cId = \{p \in Id \mid \forall x, c : (p = out(x, c)) \notin Str\}$ that are determined by the environment. Ctx consists of instances of the form $value(p) = v$ where $p \in cId$ is a connection point and v is a value.

Finally, the set Obs describes the values of those connection points that are observed (measured) by the diagnostic agent. It therefore also consists of instances of the form $value(p) = v$ where $p \in Id$ is a connection point and v is a value.

A *candidate diagnosis* is a set D of instances of the predicate $mode(,)$ such that for every component $c \in C$ there is exactly one mode in $m \in M_c$ such that $mode(c, m) \in D$. A *diagnosis* is defined as follows:

Definition 1 *Let $S = (C, M, Sd, Ctx, Obs)$ be the system to be diagnosed and let $\mid\sim$ to denote the possibly limited reasoning capabilities of a diagnostic system.*[5] *Moreover, let Obs_{con}, $Obs_{\mathrm{abd}} \subseteq Obs$ be subsets of observations and let D be a candidate diagnosis. Then D is a diagnosis for S iff*

$$D \cup Sd \cup Ctx \mid\sim \bigwedge_{\varphi \in Obs_{\mathrm{abd}}} \varphi \text{ and } D \cup Sd \cup Ctx \cup Obs_{\mathrm{con}} \not\mid\sim \bot.$$

[2] This assumption is not valid in every system. We can, however, transform most systems to a system consisting components with only inputs and outputs (see for instance [3]).

[3] A connection between components is modeled by *connection point* that is shared by one or more inputs and an output. Note that a physical connection should be modeled by component.

[4] Note that we may use a single description for a class of components. Instances of this description must imply the form of description give here.

[5] I.e $\{\varphi \mid \Sigma \mid\sim \varphi\} \subseteq \{\varphi \mid \Sigma \vdash \varphi\}$.

Remark In the literature two types of diagnoses are distinguished: *consistency based* [5, 6] and *abductive* [1] diagnosis. Both can be combined into one more general diagnostic definition [2]. This latter definition is used here.

3 Multi-agent diagnosis

A knowledge distribution over multiple agents induces a division of a system S into several subsystems. In the case of a *semantical* knowledge distribution, each agent A_i makes diagnosis of a different *aspect* of the system S. An aspect defines a system S_i of S consisting of a structural description Str_i and a behavioral description Beh_i. A component $c \in C$ has a specific behavior $mode(c, m) \rightarrow \Phi_i \in Beh_{c,i}$ for each fault mode $m \in M_c$ and each aspect i. Of course, given k different aspects, $Beh_c = \bigcup_{i=1}^{k} Beh_{c,i}$ and $\models (\Phi_1 \wedge ... \wedge \Phi_k) \leftrightarrow \Phi$ where Φ is the complete (single agent) behavior of mode m.

Without loss of generality, we may assume that the value of each output of a component is completely determined by the behavior with respect to one aspect. Therefore, also the structural description and the observations are distributed based on the aspect that determines the value of an output or requires the value of an input: Str_i and Obs_i.

By distributing knowledge, i.e. Beh_i and Str_i over the agents, we must provide agents with information about the components' inputs that (i) are needed for the components' behavioral description and that (ii) are determined by aspects that do not belong to the agent's expertise. Other agents must provide the agent i with the values of these inputs. In_i and Out_i will be used to denote the connection points the values of which are provided by other agents, respectively must be passed on to other agents. Hence, $S_i = (C_i, M, Id, Sd_i, Ctx, Obs_i, In_i, Out_i)$ is a subsystem to be diagnosed the agent. A candidate diagnosis of the subsystem S_i is denoted by D_i.

The diagnosis of one agent Each agent A_i in the multi-agent system must make a diagnosis of the subsystem $S_i = (C_i, M, Id, Sd_i, Ctx, Obs_i, In_i, Out_i)$. This can be viewed a single agent diagnosis if values of the inputs and outputs of the subsystem are known. We use the set V_i to denote value assignments $value(p) = v$, with $p \in In_i$, to the inputs. V_i is the local context of the subsystem S_i that is determined by the outputs of other subsystems. We therefore extend Definition 1 to the diagnosis of subsystems.

Definition 2 *Let $S_i = (C_i, M, Id, Sd_i, Ctx, Obs_i, In_i, Out_i)$ be a subsystem to be diagnosed. Let $Obs_{\mathrm{con},i}$, $Obs_{\mathrm{abd},i} \subseteq Obs_i$ be subsets of the observations, and let V_i be a (partial) descriptions of the values of the connection points In_i. Finally, let D_i be a candidate diagnosis. Then D_i is a diagnosis for S_i iff*

$$D_i \cup Sd_i \cup Ctx \cup V_i \models \bigwedge_{\varphi \in Obs_{\mathrm{abd},i}} \varphi \text{ and } D_i \cup Sd_i \cup Ctx \cup V_i \cup Obs_{\mathrm{con},i} \not\models \bot.$$

The diagnosis of multiple agents Given multiple diagnostic agents, an important question is how the diagnoses of the agents relate to the diagnoses of a single agent that has complete knowledge of the system description and the observations. To answer this questions we assume *there are no conflicts between the knowledge of the different agents*; i.e. there always exists a diagnosis D such that: $D \cup Sd \cup Cxt \cup Obs$ is consistent. We need this assumption because single agent diagnosis requires consistent knowledge.

Proposition 1 [6] *Let $S_1, ..., S_k$ be the subsystems that make up the system S. Moreover, let D be a single agent diagnosis of S. Then $V_i = \{(value(p) = v) \mid p \in In_i,\ D \cup Sd \cup Ctx \mid\sim (value(p) = v)\}$ is the local context of S_i that is determined by the other subsystems S_j, and $D_i = \{mode(c, s) \mid c \in C_i, mode(c, s) \in D\}$ is a diagnosis of S_i.*

Proposition 2 *Let $S_1, ..., S_k$ be the subsystems that make up the system S. Moreover, let the local context V_i of S_i describe the values of connection points in In_i that must be determined by the other subsystems S_j, and let D_i be a diagnosis of S_i determined by agent A_i given V_i. Then, $D = \bigcup_{i=1}^{k} D_i$ is a single-agent diagnosis if D is a candidate diagnosis and if for every $i = 1, ...k$: $D_i \cup Sd_i \cup Ctx \cup V_i \mid\sim (value(p) = v)$ for every $p \in Out_i$, $p \in In_j$ and $(value(p) = v) \in V_j$.*

Note that a global diagnosis D is also a diagnosis of the agent A_i if an aspect i plays a role in every component $c \in C$.

The above propositions show that multi-agent diagnosis is possible. Note, however, that given a global candidate diagnosis D, predicting the values of all connection points is an NP-Hard problem [8]. When knowledge of the system is semantically distributed over the agents, often there are only a few connection points between the subsystems managed by different agents. Moreover, if the connections between subsystems do not form cycles, the distribution of knowledge over the agents does not contribute significantly to the time complexity of predicting the system's behavior given a diagnosis. Since usually, there are not many connections between different behavioral aspects of the system, in the remainder of this paper, we will assume that the prediction of the system's behavior is not an issue.

A single agent approach is based on the implicit assumption that an agent has complete and consistent knowledge of a component's behavior given its known behavioral modes. Without this assumption, a single agent cannot make a diagnosis using Definition 1. However, when knowledge is semantically distributed, this assumption need not be valid. Therefore, we must study the consequences of incomplete and incorrect knowledge on establishing a global diagnosis.

Agents with incomplete knowledge When agents look at different aspects of a component, they may not have the same detailed knowledge for every aspect. Concerning the electrical aspects of an integrated circuit for instance, an agent may distinguish many specialized fault modes for which knowledge concerning the thermodynamic aspects of the circuit is lacking. Hence, for a component c an agent A_i may only have behavioral knowledge for some of the component's fault modes $M_{c,i} \subseteq M_c$.

The lack of knowledge about a component's behavior for some fault modes raises a problem: the agents may not be able to reach an agreement. To overcome this problem an agent A_i may just assume a behavior for each fault mode $m \in (M_c - M_{c,i})$. The question is, which behaviors can validly be assumed? If the behavior of a less specific fault mode would be known, this behavior may be used. Since a set of fault modes $M_{c,i}$ always contains the normal mode nor and the least specific fault mode ab (even if no behavior of ab is known), we may assume the existence of a hierarchy of fault modes ordered with respect to specificity. We call such a hierarchy and *abstraction hierarchy*.

[6] The proofs are omitted because of lack of space.

Definition 3 *Let c be a component with M_c as its set of behavior modes. An abstraction hierarchy on M_c is a strict partial order $\succ$ defined on $M_c - \{nor\}$ where the intuitive meaning of $m \succ m'$ is that m is more specific than m' and ab is the unique least specific element in the hierarchy, i.e. for all $m \in M_c - \{nor, ab\}$: $m \succ ab$.*

A more specific mode implies a more specific description of the faulty behavior of the component. Therefore, the following requirement must hold.

> For every $m, m' \in M_{c,i}$: if $m \succ m'$, $mode(c, m) \rightarrow \Phi \in Beh_{c,i}$ and $mode(c, m') \rightarrow \Phi' \in Beh_{c,i}$, then $\models \Phi \rightarrow \Phi'$.

Moreover, we assume that for any component c, $mode(c, ab) \rightarrow \top$ holds. That is, there is no behavioral description for the fault mode ab.

Definition 4 *Let $\Phi_{i,\mathrm{nor}}$ be the normal behavior with respect to aspect i of a component c and let Cst be a set of formulas describing the physical constraints of the world.*[7]

An abstraction hierarchy is complete *iff for each a fault mode m_0, if m_0 is not a most specific fault mode, then there is a set of fault modes $m_1, ..., m_\ell$ such that $m_j \succ m_0$ for $j \geq 1$, $mode(c, m_j) \rightarrow \Phi_{i,j} \in Beh_{c,i}$ and $Cst \models (\Phi_{i,0} \wedge \neg\Phi_{i,\mathrm{nor}}) \leftrightarrow (\Phi_{i,1} \vee ... \vee \Phi_{i,\ell})$.*

The abstraction hierarchy on the fault modes defines a similar abstraction hierarchy on the diagnoses.

Definition 5 *Let D, D' be two candidate diagnoses. D is at least as specific as D', $D \succeq D'$, iff for every $mode(c, m) \in D$ there is a $mode(c, m') \in D'$ such that $m \succeq m'$.*

Note that agents that wish to give a best possible explanation for the observed anomalies, should focus on the *most specific* diagnosis. Whether the agents only determine the most specific diagnoses depends on the type of diagnosis they use; i.e. the choice for the sets Obs_{abd} and Obs_{con}.

Proposition 3 *Pure abductive diagnosis produces only the most specific diagnoses.*

Pure consistency based diagnosis also returns every less specific diagnosis.

Proposition 4 *Let $S_1, ..., S_k$ be the subsystems that make up the system S and let the abstraction hierarchy of fault modes be complete. Moreover, let D be a most specific diagnosis of S. Then there exists a set of most specific diagnoses $D_1, ..., D_k$ for respectively $S_1, ..., S_k$ such that $D = \bigcup_{i=1}^{k} D_i$.*

The behavioral description $Beh_{c,i}$ of a component with respect to an aspect need not specify a behavior for each fault mode in M_c. In order for the agent to establish a global diagnosis, the missing behaviors have to be added. The following assumption serves this purpose.

Assumption *A fault mode m of a component c for which an agent has no behavioral knowledge, has the same behavior as the most specific mode $m' \in M_{c,i}$ such that $m \succeq m'$.*

The assumption extends the behavioral description, making the behavioral knowledge of every fault mode of every component complete for all aspects. Hence, the results of propositions 1 and 2 apply.

[7] Some abnormal behaviors of a component need not be physically possible. For these behaviors, a complete hierarchy need not contain a fault modes describing them.

Agents with incorrect knowledge Agents lacking knowledge about behavior modes is not the only problem that may arise in a multi-agent system. Knowledge of agents may in some situation be incorrect leading to inconsistencies between local diagnoses. Hence, the agents will not be able to agree on a global diagnosis.

A robust multi agent system should be able to handle situations in which inconsistencies between local diagnoses arise. One possibility, which has been proposed in [10], is the use of voting. However, if agents look at different aspects of the system, voting offers no solution. Moreover, voting requires the communication of all local diagnoses of all agents. The number of these diagnoses may be exponential in the number of components.

The abstraction hierarchy on the fault modes also makes it possible to handle inconsistencies. When agents cannot agree on a most specific diagnosis, they may investigate whether they can resolve the inconsistency by looking at less specific diagnoses. If the agents apply pure consistency based diagnosis, such a diagnosis always exits since there is no behavioral description for the fault mode ab. Hence, one or more global diagnoses always exists but these global diagnoses need not correspond with most specific diagnoses of individual agents. An agent may determine several more specific diagnoses which cannot be diagnoses according to other agents. Especially if the abstraction hierarchy of fault modes is complete, knowledge of the agents must be inconsistent.

Proposition 5 *Let $S_1, ..., S_k$ be the subsystems that make up the system S and let the abstraction hierarchy of fault modes be complete. Moreover, let D be a most specific diagnosis of S. Then, the knowledge of two agents A_i and A_j is inconsistent if agent A_i has a diagnosis $D_i \succ D$ and for every diagnosis $D_i' \succ D$ there exist no diagnosis D_j established by agent A_j such that $\{mode(c, m) \in D_i' \mid c \in C_j\} \subseteq D_j$.*

A difficult issue is comparing the quality of the diagnoses. Clearly, the most specific diagnoses are preferred. There is, however, another way in which diagnoses can be distinguished. Assuming that the abstraction hierarchy of fault modes is complete, given two consistency based diagnoses D and D', the diagnosis D may abductively explain an observation φ while a diagnosis D' may not. Clearly, diagnoses that give an better explanation should be preferred.

Definition 6 *A diagnosis D gives a better explanation than diagnosis D' iff* $\{\varphi \in Obs \mid D' \cup Sd \cup Cxt \mid\sim \varphi\} \subseteq \{\varphi \in Obs \mid D \cup Sd \cup Cxt \mid\sim \varphi\}$.

4 Protocols for establishing a diagnostic agreement

The agents may determine a global diagnosis by first determining all fault modes $M_c = \bigcup_{i=1}^{m} M_{c,i}$ as well as the abstraction hierarchy $\succ$ on M_c for each component c, and subsequently exchanging all their local diagnoses. The first step is straight forward and will not be discussed here because of space limitations. The second step is more problematic. The number of diagnoses to be exchanged between the agents can be quite high and can be exponential in the number of component is the worst case. In order to control the complexity, agents should focus on diagnoses in which a minimal number of components, with respect to $\subseteq$, are broken.

Since a local minimal diagnosis need not be a global minimum diagnosis, the agent proposing the diagnosis needs to receive feedback when a proposed diagnosis is rejected

by other agents. Subsequently, the agent can generate a new diagnosis taking into account the diagnoses that have been rejected.

The generation of new minimal diagnoses can be improved if agents supply the reasons for rejecting a proposed diagnosis. When agent A_1 proposes a partial diagnosis D_1, agents $A_2, ..., A_k$ might reject the diagnosis because some (combination of) modes is inconsistent with its observations. Let $R_i \subseteq D_1$ be such (a combination of) modes. Then R_i is a smallest subset of D_1 such that: $R_i \cup Sd_i \cup Ctx \cup Obs_i \mid\sim \perp$ for $2 \leq i \leq k$.

Note that an agent A_i might determine more than one smallest subset R_i. If SR_i is the set of all smallest subsets R_i, agent A_1 can use this information $TR = \bigcup_{2 \leq i \leq k} SR_i$ as a set of constraints in its search for a next diagnosis. It may not select a new diagnosis D_1' containing any $R_i \in TR$ as a subset.

The following simple protocol shows how the agents may proceed. To gain robustness, eventually, always one of the agents takes the initiative to establishes the global diagnoses. In the protocol, the agent that takes the initiative is agent A_1.

Agent	**Action**
A_1	$TR := \varnothing$;
A_1	finished := false;
A_1	while not finished do
A_1	generate the next most specific minimal diagnosis D_1 of S_1
	such that for no $R \in TR$: $R \subseteq D_1$;
A_1	finished := not diagnosis_found;
A_1	while diagnosis_found, for $i := 2$ to k do
A_1	send 'propose D_1' to A_i;
A_i	receive 'propose D_1' from A_1;
A_i	determine a most specific local diagnosis D_i of S_i such that $D_1 \succeq D_i$;
A_i	if a diagnosis D_i exists then;
A_i	send 'accept D_i' to A_1;
A_i	else
A_i	send 'reject SR_i' to A_1;
A_i	end;
A_1	if received 'accept D_i' from A_i then
A_1	$D_1 := D_i$;
A_1	else if received 'reject SR_i' from A_i then
A_1	$TR := TR \cup SR_i$;
A_1	diagnosis_found := false;
A_1	end;
A_1	end;
A_1	if diagnosis_found then
A_1	store D_1;
A_1	end;
A_1	end;

5 Conclusion

In this paper, we analyzed the problem of multi-agent diagnosis when knowledge is semantically distributed over the agents. Especially the case that the agents' knowledge concerning the faulty behavior of some components, is incorrect has been considered. A solution based on an abstraction hierarchy on the fault modes has been proposed and a protocol for determining the global diagnoses with a minimal number of broken components has been given.

An important question for further research is how to order the global minimal diagnoses. A minimal diagnosis in which the knowledge of the agents can be assumed to be correct gives a better explanation of the observed anomalies than a diagnosis that is less specific in order to deal with incorrectness in agents' knowledge. It is not obvious, however, how to order diagnoses implying that the knowledge of some agent cannot be correct in the current situation. Should, for instance, a diagnosis in which all agents assume that a component is broken though they do no agree on the fault mode, be better than a diagnosis in which some agents assume the component to be broken and agree on th fault mode and the other agent assume the component is not broken?

References

[1] L. Console and P. Torasso. Hypothetical reasoning in causal models. *International Journal of Intelligence Systems*, 5:83–124, 1990.

[2] L. Console and P. Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7:133–141, 1991.

[3] J. J. v. Dixhoorn. Bond graphs and the challenge of a unified modelling theory of physical systems. In F. E. Cellier, editor, *Progress in Modelling & Simulation*, pages 207–245. Academic Press, 1982.

[4] P. Frohlich, I. de Almeida Mora, W. Nejdl, and M. Schroeder. Diagnostic agents for distributed systems. In J.-J. C. Meyer and P.-Y. Schobbens, editors, *Formal Models of Agents. ESPRIT Project ModelAge Final Report Selected Papers. LNAI 1760*, pages 173–186. Springer-Verlag, 2000.

[5] J. d. Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[6] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[7] N. Roos, A. ten Teije, A. Bos, and C. Witteveen. Multi-agent diagnosis: an analysis. In *BNAIC'01*, 2001.

[8] N. Roos, A. ten Teije, A. Bos, and C. Witteveen. An analysis of multi-agent diagnosis. In *AAMAS 2002*, 2002.

[9] N. Roos, A. ten Teije, and C. Witteveen. A protocol for multi-agent diagnosis with spatially distributed knowledge. In *AAMAS 2003*, 2003.

[10] M. Schroeder and G. Wagner. Distributed diagnosis by vivid agents. In *Proceedings of the first conference on Autonomous Agents*, pages 268–275, 1997.

Problem solving in a computational society

Nico Roos [a] Cees Witteveen [b]

[a] Universiteit Maastricht, IKAT, P.O.Box 616, 6200 MD Maastricht.
[b] Delft University of Technology, ITS, PO Box 5031, 2600 GA Delft.

Abstract

We propose a new paradigm for solving problems in a computational society where a computational agent is able to use information about the direct and indirect computational experiences of other agents. We call this framework the history computing (HC) framework and use it to generalize existing computational frameworks that might be used in a computational agent society, like relativized or oracle computing and knowledge compilation (KC). We present a classification of problems using this framework and give some examples showing that direct compilation of computational experiences of other agents can be significantly enhanced by adding meta-information about these experiences. Finally we discuss some relationships with relativized computing and KC, showing that the HC-framework is able to refine both.

1 Introduction

Traditionally, complexity analysis considers problem solving as an isolated *one-shot* activity, taking into account only time and/or space resources of algorithms needed to solve a problem. In this paper, we will consider the computational complexity of a problem as not only dependent upon the time and space resources required but also upon past *computational experiences*. These computational experiences originate in a computational environment of computing devices (a computational society) having solved several instances of problems (cf. [6]). Together, these computational experiences can be conceived as a *collective store* of instances of computational problems and their solutions. We will call the direct results of such computational experiences a *computational history*. Such a history H contains a concise description of the instance of some problem solved and its solution. Further, (meta-)information about the history can be given in the form of a concise description $\Phi(H)$ of the properties of the history. The purpose of this paper is to show that reasonable amounts of information about such computational experiences (history and meta-information) can alleviate the computational costs of solving instances of the same or a related target problem.

The organization of this paper is as follows. First, we present the *History Based Computing* (HC) framework. Then we present some examples used to give a classification of the complexity of some problems using the framework. Next, we discuss some related approaches and we show that it refines relativized computations, and we show that (i) the HC-framework is equivalent in computational power to a direct generalization of the Knowledge Compilation (KC)-framework presented by Cadoli et al (cf. [4]), but (ii) it refines this framework in a considerable way by distinguishing between the direct computational power of the *individual* agents in the computational environment and the

meta-information that has to be provided by the *collection* of agents. Finally, we point out some directions for future research.

2 History Based Computing: basics

A computational history H is a collection of problem instances (of some not necessarily identical problems) and their solution produced by a computational agent society. Formally, a *computational history* is a set $H = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ of tuples $\sigma_i = (x_i, s_i)$ where each x_i is an instance of some problem P_i and s_i is its solution[1]. The *size* of a history $||H||$ is the total number of symbols occurring in the history H.

We are especially interested in solving instances of *intractable* problems using modest, *polynomially sized*, computational histories. To model efficient computations using such histories we consider polynomial algorithms that are capable to solve instances x of a problem P by inspecting a computational history H_x assigned to x. These algorithms look look for instances x_i and their solution s_i occurring in H_x and use these solutions to find a solution for x. We assume the retrieval of the answers to x_i using H to cost polynomial time. Since we will not assume H_x to be ordered, we will therefore simply require $||H_x||$ to be polynomially bounded in the length $|x|$ of x.
Except for collecting histories we assume a computational agent society also to be able to provide agents with relevant *meta-information* about such histories. Such meta-information about a history H, given in the form of a sentence $\Phi(H)$, often enables us to use more compact histories and also can be used to guarantee the correctness of the algorithm used by the agent.
For example, suppose that an agent uses a database D and wants to know whether some literal[2] $l \in Lit(D)$ is true given D. Now suppose that in the agent society numerous literal consequence queries have been posed and only a small fraction $S = \{l_1, \ldots, l_n\}$ of the literals turn out to be logical consequences of D. Then, instead of presenting all positive instances $D \models l$ together with all negative instances $D \not\models l$ (a possibly huge set), the agent society could specify their computational experience in a more succinct way by presenting the set S together with a sentence $\Phi(S) \equiv \forall l \in Lit(D)[(D \models l) \Leftrightarrow (l \in S)]$ expressing that this set of computational experiences contains *all* the literals in the set $Lit(D)$ that are logical consequences of D.

Algorithmically, such a sentence $\Phi(H)$ serves as the *precondition* of an algorithm A that, given an instance x and a history H as input, computes an answer y. That is, $\Phi(H)$ is the *precondition* that has to be satisfied to deduce the Hoare triple $\langle \Phi(H) \{ A(x, H) \} (y = s(x)) \rangle$, where $y = s(x)$ specifies the postcondition that y is the solution to x. We will always assume the length $|\Phi(H)|$ of $\Phi(H)$ to be polynomial in $||H||$. If $\Phi(H) = \mathbf{true}$, the meta-information is called trivial and will be omitted.

Note that, without further constraining the use of such histories H, every problem could be solved in polynomial time using a computational history[3]. There are, however,

[1] Although in most cases we will deal with decision problems only, the problems P_i might also be search problems. In that case, we will always assume that the length $|s_i|$ of the solution s_i is polynomially bounded in the length $|x_i|$ of the instance x_i.

[2] $Lit(D)$ denotes the set of all literals over the database D considered as a propositional theory

[3] For every x, just consider a computational history $H_x = \{x, s_x\}$ and an algorithm inspecting H_x to find

at least two kinds of restrictions that can be used to make history based computing (abbreviated by *h-computing*) non-trivial:

1. Putting restrictions on the computational power of the agents, i.e., restricting the *complexity* of the problems whose instances are solved by the agents in the computational society.

2. Requiring that, given a history H and possibly some meta-information $\Phi(H)$, not just one single instance but a (super-polynomial) *subset* of instances of the target problem can be solved efficiently using the same history H.

The first restriction implies imposing restrictions on the class of problems from which instances occurring in H are selected. Given a history H, we therefore define C_H, or just C, as the complexity class containing all those problems whose instances occur in H. In this case, H is said to be a *history over* C. To meet the second requirement, we specify a partition on D_P, the domain (i.e., the set of instances) of P, such that every instance in a block of this partition uses the same history to find a solution. More precisely, let $h^* : D_P \rightarrow 2^{D_P}$ be a function that assigns to each instance $x \in D_P$ a set of instances y and satisfies the partitioning condition $\forall x, y[y \in h^*(x) \leftrightarrow h^*(x) = h^*(y)]$. Intuitively, if there exists a history H used to solve x in polynomial time, the same history should be used to solve every $y \in h^*(x)$ in polynomial time. We call h^* *super-polynomial* if $|h^*(x)|$ is not polynomially bounded in $|x|$. We now formally define a pair (P, h^*) consisting of a problem P and a partitioning function h^* to be h-computable as follows:

Definition 1 *A pair (P, h^*), where h^* defines a partition on D_P, is called* h-computable *if there exists a set $\mathcal{H}$ of histories, a formula $\Phi(z)$ with a free variable z, an algorithm A and polynomials $p(), q()$ such that for every $x \in D_P$*

1. *there exists a history $H \in \mathcal{H}$ such that $||H|| + |\Phi(H)| \leq p(|x|)$;*

2. *for every $y \in h^*(x)$, the algorithm A, taking y and H as input, is a correct algorithm for computing the solution $s(y)$ in $q(|y|)$-time, using $\Phi(H)$ as precondition.*

First of all, we want to investigate (new) properties of this framework that enable us to relate the computational power of individual agents in an agent society, the information the computational society as a whole can offer and the capabilities of the agent trying to solve a target problem. In particular, in this paper we will concentrate on (a) the possibilities the individual agent has to (partially) construct and check the history given to it and (b) the role of the additional meta-information provided.

3 A classification of computational problems

In classifying h-computable problems, first of all we concentrate on the use of histories where the only trivial meta-information, i.e. $\Phi(H) = \mathbf{true}$, is needed. Within this use of bare histories, a weakest form is the use of the history as an answering mechanism to queries (problem instances) the agent himself is able to construct. We call such histories *polynomially constructible*.

---

the solution for x.

If an agent is *not* able to *construct* the queries, but still does not need (non-trivial) meta-information to *check* that the history given provides sufficient information to be used by its algorithm, we say that the history is *polynomially checkable*. Clearly, polynomially checkable histories are also polynomially constructible.

The strongest form of dependency (on the environment) occurs if the agent is dependent upon the environment not only with respect to the history H provided, but also with respect to additionally provided meta-information $\Phi(H)$ about the history such that H and $\Phi(H)$ are both needed to check that the algorithm can be used to solve instances of the target problem. We call the class of such problems $\Phi(H)$*-checkable histories*.

Below we present a more precise description of these classes together with some examples.

Polynomially constructible histories

Suppose an agent wants to solve some instance x of a problem P and is capable to generate (in polynomial time) a set of instances whose solution would enable it to compute the solution to x and other instances in polynomial time. Hence, the computational environment has only to provide the agent with the answers to a polynomial number of queries and to compose a history H containing the instances and their solutions. There is no need for the agent society to provide additional information about the properties of H, i.e., the role of the society as a coordinating entity can be neglected. After receiving the history H, the agent should be able to compute the solution to x and all $y \in h^*(x)$ in polynomial time. We call such a problem (P, h^*) a *strongly h-computable problem*. The class of strongly h-computable problems is denoted by $\mathbf{P}^s_{hist}$. Likewise, the class $\mathbf{P}^s_{hist}[C]$ is the class of strongly computable problems using histories over C.

Example 1 The conjunctive literal inference problem (CLC) (cf. [1]) is stated as follows: Given a theory T and a conjunction ϕ of literals over $atoms(T)$, is ϕ a logical consequence of T? To solve an arbitrary ϕ in polynomial time, the agent has to ask for each atom $a \in atoms(T)$, whether or not $T \models a$ and/or $T \models \neg a$. The history consisting of these $(2n)$ instances together with their answers enables the agent to answer every conjunctive literal query $\phi = \bigwedge_{i=1}^{n} l_i$ in polynomial time by a simple (history) look-up as $T \models \bigwedge_{i=1}^{n} l_i$ iff $T \models l_i$ for $i = 1, \ldots, n$.

Note that in this example, the computational environment requires the existence of simple (co-)NP-oracles (agents) that are able to solve the literal consequence problem[4] There is no need to provide non-trivial meta-information $\Phi(H)$ since (assuming the correctness of the answers) the agent is able to check the completeness of the history. In fact, the agent is able to specify all the instances (T, l) that have to occur in the history. Finally, note that a *superpolynomial* number $(O(2^n)$ queries ϕ can be answered after a history of $2n$ queries has been given. Hence, $(CLC, h^*) \in \mathbf{P}^s_{hist}[NP]$ for even superpolynomial h^*.

Polynomially checkable histories

Suppose that an agent is able to solve instances x of some problem P with some history

[4]Unlike standard oracle-computing, however, the answers of the oracles are stored and used to answer more than one single query. See also Section 4.

H such that *(i)* once the history is given, the agent is able to check in an efficient (= polynomial) way that H indeed contains sufficient information to guarantee the correctness of its algorithm, but *(ii)* is not able to generate the instances that have to occur in the history. In this case the agent society has to come up with such a history H. Such a problem is called *simply h-computable*. The class of such problems is denoted by $\mathbf{P}_{hist}$. Likewise, we define $\mathbf{P}_{hist}[C]$.

Example 2 Suppose we have an agent society consisting of satisfiability (SAT) oracles that are able to decide whether or not a propositional theory is satisfiable. Let T be a given (satisfiable) theory and C a clause (i.e. a disjunction of literals). Furthermore, for an interpretation I of T, C_I is the conjunction of all literals true under I. We have to decide whether or not $T \models C$ and, if C is not a logical consequence of T, to come up with a counter-model, The agents in the SAT-society, proceed as follows: Let $\mathcal{C} = \{C_I \mid I \text{ is an interpretation of } T \text{ and } C_I \text{ has no literal in common with } C\}$. Each agent takes a (different) clause C' from $\mathcal{C}$ and tests the satisfiability of $T \wedge C'$. If an agent finds its instance to be a yes-instance, the history $H = \{(T \wedge C'), yes)\}$ will consist of one arbitrarily chosen yes-instance. Else, the history H will contain the single no-instance $(T \wedge \neg C, no)$. Note that $T \not\models C$ iff there exists a model M of T that falsifies C, i.e., an interpretation M such that does M not have any literal with C in common and $T \wedge M$ is satisfiable. If such a counter model does not exist, $T \wedge C$ must be unsatisfiable. In both cases, the target agent is able to check in polynomial time that the history H provided is sufficient to produce the answer that C is a logical consequence of T or to provide a counter-model M for C.

$\Phi(H)$-checkable histories

Sometimes the agent society has to provide additional information about the history H given to an agent in order to guarantee that the polynomial algorithm used by the agent to solve instances of a target problem computes correct results. Such information should be *compact*, i.e., polynomial in the length of the history provided. This information might be the result of difficult and lengthy computing by the agent society, for example, by investigating an exponential amount of computational experiences, but should be captured by a succinct statement $\Phi(H)$.

Such a problem will be called *extended h-computable*. We call the class of this problems $\mathbf{P}^{\Phi}_{hist}$. Likewise, we define $\mathbf{P}^{\Phi}_{hist}[C]$ for histories over C.

Example 3 Minimal model reasoning is an important topic in artificial intelligence. We present an example using the General Closed World Assumption (abbreviated GCWA) (cf. [3, 5]). Given a theory T, the closure of T under the GCWA-rule equals $GCWA(T) = T \cup \{\neg p \mid p \in atoms(T)$ is false in every minimal model of $T\}$. The GCWA-*inference* problem (abbreviated GCWINF) is the problem "Given T and a formula ϕ, decide whether $GCWA(T) \models \phi$". The GCWA-*model checking* problem (GCWMC) is the problem "Given T and an interpretation M for T, decide whether $M \models GCWA(T)$". It is not difficult to show that an agent having to solve the GCWMC-problem can profit from access to GCWINF-agents: just ask these agents to solve GCWINF-instances $(T, \neg a)$ for every $a \in atoms(T)$. Collecting the results, the agent is able to construct $GCWA(T)$ and then the problem $M \models GCWA(T)$ is solvable in polynomial time for arbitrary

interpretations[5] M. The GCWINF-problem, however, is a rather hard[6] problem. So, although there is a complexity class $C = \Sigma_3^p$ such that $(GCWMC, h^*) \in \mathbf{P}^s_{hist}[C]$, we could imagine that there are no GCWINF-agents in the environment and we have only available very simple agents (but also very many of them) that test whether M is a model for T. We can solve the GCWMC-problem by using (an exponential) number of simple model-checking agents: Each agent is given an interpretation I of T such that every interpretation is checked by at least one agent. The agent society collects all possible yes-instances $(T, M, "yes")$, compares the models and selects the minimal ones. Finally, for every $a \in atoms(T)$ *at most* one minimal model M_a is selected verifying a (i.e. $M \models a$) and, if such a model exists[7], the triple $(M_a, T, "yes")$ is added to the history H. The meta-information about this polynomial history H is expressed by the following sentence:
$\Phi(H) \equiv \forall a \in atoms(T)[GCWA(T) \not\models \neg a \Leftrightarrow \exists M[(T, M, "yes") \in H \ \& \ M \models a] \ ]$
Now the target agent can use a simple algorithm to check whether an arbitrary interpretation I is a GCWA-model of T: first, check (in polynomial time) whether $I \models T$. If so, for every atom a occurring in I, check whether the history H contains a verifier M_a for a. If all tests have been passed, I is a GCWA-model of T, else it is not. Note that $\Phi(H)$ is essential for this testing procedure: it guarantees that if there exists a verifier for an atom a it does occur in the history.

4 Some comparisons and results

Separation results We present some preliminary results. It should come as no surprise that in general we have $\mathbf{P}^s_{hist} \subseteq \mathbf{P}_{hist} \subseteq \mathbf{P}^{\Phi}_{hist}$. But in fact, we can easily prove the equality of these classes in *their unrestricted* versions: $\mathbf{P}^s_{hist} = \mathbf{P}_{hist} = \mathbf{P}^{\Phi}_{hist}$.

Only if we relativize the computational powers of the individual agents in the agent society, we can prove some separation results. To give an example, note that in the last example (Example 3) in fact we showed that $(GCWMC, h^*) \in \mathbf{P}^{\Phi}_{hist}[P]$ since we only need individual agents to perform propositional model-checking which is in P . The GCWMC-problem, however, is known to be an NP-Hard problem. Hence, it follows immediately that $(GCWMC, h^*) \in \mathbf{P}^s_{hist}[P] = P$ implies $P = NP$. Hence, there are complexity classes C such that $\mathbf{P}^s_{hist}[C]$ is strictly contained in $\mathbf{P}^{\Phi}_{hist}[C]$ unless $P = NP$.

Unless some major assumption concerning standard complexity classes do not hold, we can state:

Proposition 1 *There exist complexity classes C, C' such that $\mathbf{P}^s_{hist}[C] \subset \mathbf{P}_{hist}[C]$ and $\mathbf{P}_{hist}[C'] \subset \mathbf{P}^{\Phi}_{hist}[C']$.*

Relations with other approaches There are two approaches that are closely related to history computing. A closer inspection to their relationships with HC-computing also shows that the latter significantly refines the complexity classes characterizable in both approaches.

[5] Note that $O(2^n)$ instances (M, T) can be solved using this computational history.
[6] It is known to be Π_2^p-hard and to be in Σ_3^p
[7] Note that such a model indicates that $GCWA(T) \not\models \neg a$.

Oracle computing Oracle computing or *relativized computational complexity* addresses the problem how difficult certain (target) problems would be if other problems could be solved at $O(1)$-cost by consulting *oracles* for these other problems. The class of problems solvable in polynomial time using oracles for problems in C is denoted by $\mathbf{P}^C$. History-based computing (in the strong computable sense) strictly refines oracle computing[8] in the sense that taking h^* to be the identity function we have $\mathbf{P}^C = \mathbf{P}^s_{hist}[C]$, but the use of history functions allows us to make finer distinctions within this class[9]. It is only with respect to superpolynomial partition functions that we can refine $\mathbf{P}^C$: For example, both the conjunctive literal consequence problem and the general formula consequence problem (Given a theory T and a formula ϕ, is ϕ a logical consequence of T) belong to $\mathbf{P}^{NP}$, but only the former allows for the existence of super-polynomial partitions in $\mathbf{P}_{hist}[NP]$. The general formula consequence problem with a superpolynomial h^* defined as $h^*(T, \phi) = \{(T, \psi) \mid \psi \in \mathcal{L}_T\}$, however, does not occur[10] in $\mathbf{P}_{hist}[NP]$ unless $P = NP$. Also note that $\mathbf{P}_{hist}\Phi[C]$ strictly contains $\mathbf{P}^C$ as in the former the agent itself does not need to construct the instances of the history.

Secondly, the use of meta-information can greatly enhance the computational power of relativized history-based computing: There are complexity classes C and formulas $\Phi(H)$ such that $\mathbf{P}^C \subset \mathbf{P}^{\Phi}_{hist}[C]$ where the inclusion is strict.

Knowledge compilation Knowledge Compilation (KC) concentrates on the question how we can reduce computational resources to solve a problem if every instance z of a problem can be split up in a *fixed* part x (i.e., that piece of the problem instances that remains constant considering a set of instances) and a *variable* part y. Instead of using the fixed part x directly, x is *compiled* into a polynomially sized structure $d(x)$ and, instead of solving instances of the form $z = (x, y)$, instances $z' = (d(x), y)$ are solved. *On-line* computational efforts may be reduced if, after such an *off-line* compilation phase, a more efficient algorithm can be used that is able to access the pre-processed information (i.e., the common part $d(x)$) needed to answer future queries for which only the variable part y is of interest. The computational effort of this algorithm is measured only in terms of the on-line costs. If F is the set of fixed parts and V the set of variable parts of a problem P, this problem is said to be *KC-compilable* iff for every $x \in F$ there exists a polynomially sized, with respect to $|x|$, data structure $d(x)$ such that for every variable part $y \in V$, an instance (x, y) of P can be solved in polynomial time using $d(x)$ instead of x (cf. [2, 1]).

There is a natural embedding of such a fixed-variable part distinction of KC and the use of our history function h: Let (P, F, V) be a problem with fixed parts F and variable parts V. let $f_P(z) \in F$ denote the fixed part of the instance $z \in D_P$. We define the partition function $h^*_{kc}(\cdot)$ associated with P as $h^*_{kc}(x) = \{y \ : \ f_P(x) = f_P(y)\}$ for every $x \in dom(P)$. It turns out to be easy to prove that KC-compilable problems are also h-computable: If $(P, F, V) \in \mathbf{P}_{kc}$ then $(P, h^*_{kc}) \in \mathbf{P}_{hist}$.

KC can be easily generalized by using more general partition functions h^* than h^*_{kc}. Let us denote by $\mathbf{P}_{\mathbf{kc}^*}$ the class of KC-compilable problems using general partition functions. The following result states that this generalization exactly equals the class of

[8] Note that oracle computing in our sense does not allow interaction between the oracles and the computing agent during the computation of the latter. Therefore, we restrict ourself to *non-adaptive* queries to oracles.

[9] In fact, for every polynomial h^* , we have $P \in \mathbf{P}^C$ iff $(P, h^*) \in \mathbf{P}^s_{hist}[C]$.

[10] This result follows immediately from the correspondence with KC and occurs in [2].

history-computable functions:

Proposition 2 $(P, h^*) \in \mathbf{P}_{\mathbf{kc}^*}$ *iff* $(P, h^*) \in \mathbf{P}_{hist}$.

The previous proposition essentially states that the restriction of the compiled KC datastructure to a history i.e., a set of instances of problems together with their solutions does not constitute a restriction. In fact this equivalence enables us to claim that the KC-framework in fact is too general, since it completely neglects the computational resources needed to construct the datastructure, whereas the HC-framework recognizes these differences by pointing out *(i)* the complexity class where the problems whose instances occur in the history belong to and *(ii)* distinguishing between these results of individual problem solving activities and the results of cooperation between the computing agents needed to provide additional meta-information.

5 Conclusions and further research

This paper is a preliminary investigation into the analysis of multi-agent computing and a framework for studying the complexity of those computations. Future research should concentrate on the following themes: *(i)* the relationship between the problem solving capacities of the individual agents and properties of the meta-information Φ about the results of those computations; *(ii)* the relationship between the problem solving capacities of the individual agents and the size of the subset of instances of the target problem solvable by a polynomial agent; *(iii)* introduction of (limited) interaction during the problem solving process and its consequences; for example what happens if we allow adaptive queries to be raised and how should this influence the problem solving capability of the agent solving instances of the target problem.

References

[1] M. Cadoli and F. M. Donini. A survey on knowledge compilation. *AI Communications*, 10:137–150, 1997.

[2] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Feasibility and unfeasibility of off-line processing. In *ISTCS 96*, pages 100–109, 1996.

[3] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Space efficiency of propositional knowledge representation formalisms. *Journal of Artificial Intelligence Research*, 13:1–31, 2000.

[4] M. Cadoli, F. M. Donini, and M. Schaerf. Is intractability of nonmonotonic reasoning a real drawback? *Artificial Intelligence*, 88:215–251, 1996.

[5] S. Coste-Marquis and P. Marquis. Knowledge compilation for circumscription and closed-world reasoning. *Journ. of Logic and Computation*, 11(4):579–607, 2001.

[6] S. Sen and G. Weiss. *Learning in Multiagent Systems*, pages 259–298. The MIT Press, Cambridge, 1999.

Learning an approximate map of the environment by unsupervised bimodal landmark exploration

Lambert Schomaker [a] Rudolf Fehrmann [a]

[a] AI Inst./University of Groningen, Grote Kruisstraat 2/1, 9712 TS Groningen, The Netherlands

Abstract

Technological methods for navigation have achieved a high level of perfection, currently. However, this perfection is mostly achieved at the cost of additional artificial equipment which is extraneous to a freely-navigating agent such as an autonomous robot. Cognitive, biological navigation is based on functionality which is intrinsic to the cognitive system and thus more flexible and autonomous in the true sense. In this paper, exploratory experiments in navigation are performed, based on proximity events, visual landmarks, and distances traveled. It will be shown that a robot is able to learn a Kohonen self-organized landmark map of image and sonar data. An approximate 2-D map of the environment can be computed on the basis of the two major principal components within a sparse distance matrix between landmarks.

1 Introduction

Technological systems for navigation, i.e., satellite-based global positioning (GPS) and laser-based triangulation have achieved a level of utmost perfection. Also, artificial sensing such as 2-D laser range finding may provide stable information for navigational mapping [8]. However, it is amazing how good animals are at navigating through their habitat, without using additional hardware, laser, absolute coordinate systems or rectangular-grid maps. On the contrary, in biological systems, ego-centric and landmark-based coding play a predominant role, yielding navigation behavior which statistically reveals an underlying robust and flexible mechanism which supports survival in a real environment. Therefore, it remains a scientific challenge to develop models for map learning which are based on biologically plausible representations and learning schemes. The current work is intended to be a basic step in the direction of robust, cognitive navigation modeling. This is opposed to the traditional paradigm, stemming from the 1980s, where exact geometric "wire-frame" models of a particular environment are designed by the human supervisors and transferred to a robot's navigation subsystem. It became clear that the unavoidable discrepancies between an internal environment model and the actual world cannot be solved elegantly within this approach. Since realistic environments are not static there is a need for flexible and adaptive navigation methods such as can be found in biological systems. We will therefore postpone the use of absolute Cartesian grids and focus on (1) an approximate navigation modeling system which (2) can ultimately be combined with a definition of behaviors which allow for an escape from local navigation problems. In

the following paragraphs the basic mechanisms of navigation in relative simple organisms such as bees, ants and rats will be discussed, incorporating results at the neural and behavioral level.

1.1 Animal navigation

Dead Reckoning, the first mechanism which seems to be involved in animal navigation concerns the continual updating of position and heading by summing successive small ego displacements in the environment. Such an integration of the velocity vector with respect to time will yield an estimate of the position vector. Using this method, the animal is capable to keep track of its current position and directional heading in the environment when there is no perceptual input available to serve as a cue on which it can determine its current position and directional heading. At the neural level, two types of neurons in the rat brain have been implicated in spatial learning and navigation processes: the so called *place cells*[6, 5] of the hippocampus and *head direction cells* of the thalamus and postsubiculum.

Landmark Orientation. Another prevailing mechanism in animal navigation is landmark based. An animal may use the perceptual pattern of familiar, distinctive and easily perceptible points in the environment to determine its current directional heading and position in the environment. The available perceptual input allows for an association with a learned perceptual pattern of a landmark perceived earlier, activating the relative positions of that landmark in the environment to other nearby landmarks which were visited during the learning history. The resulting network of associations is called a cognitive map (Tolman[9]), which in its representation strongly differs from Cartesian grid maps. The use of landmarks orientation is a useful addition to dead reckoning mechanisms, since the detection of a landmark allows for resetting the accumulated position and heading error on the basis of the known and encountered landmark. Wehner and Raber[7] conducted experiments done on desert ants *Cataglyphis bicolor* that suggest that in order to move to a particular location ants try to match their current perceptual input in the form of a visual percept, with stored visual inputs of landmarks. Cartwright and Collet[1] conducted similar experiments on bees, with similar results. At the neurological level of navigation, Muller and Kubie[5] conducted an experiment with rats which provided evidence that *place cells* are also capable of making use of visual landmarks in the environment to establish a place-specific firing pattern.

Other mechanisms for biological navigation exist, such as beacon navigation and gradient-based navigation. These are treated elsewhere[3].

1.2 A simplified model

We will start to explore the concept of cognitive robot navigation based on navigational principles found in relative simple organisms, such as insects, using a Pioneer-II robot. The ultimate end result in our approach will yield a system which is capable of navigation on basis of a landmark-based map of the environment that is being constructed incrementally while the robot is exploring the environment. The navigation model will have to be based on the following components: (a) Multi-modal Landmarks, (b) Dead reckoning and (c) Cognitive maps.

A multi-sensor definition of landmarks. A fundamental characteristic of a landmark is that it is clearly identifiable within the ongoing perceptual stream, and sufficiently

unique in the environment to be a basis for navigation. Possibly, a salience feature can be extracted, which indicates landmark candidates in the perceptual stream. It seems likely that such mono-modal perceptual mechanisms do indeed exist (e.g.: a highly satiated color of a flower). However, assuming that landmarks are in some ecological way relevant to the species at hand, it seems likely that multi-sensor input will play a role in landmark determination. Notably, we introduce here the notion of *proximity event*. It is assumed that landmarks are specific samples of multi-sensory input which refer to a location in the environment where 'egocentric space' is invaded by physical objects which are relevant to the interests of the autonomous agent. Proximity events are important because they may indicate imminent collision and/or the presence of Friend/Foe or Food (FFF). Colby and Goldberg[2] describe neurons in the medial intraparietal(MIP) area specialized for responding to stimuli (in our case: potential landmarks) that invade this 'egocentric space'.

Dead reckoning. Dead reckoning is a vulnerable numerical integrator mechanism in many computational substrates. In some animals, optic flow may be the basis of ego-velocity estimation. For the experiments, we could only make use of the Pioneer-II odometry, which appeared to be highly biased by wheel slip. Heading information is especially unreliable, and no vestibular sensor is present to determine actual achieved angular velocity. However, it appeared that the total length of the approximately linear path traveled between two points can be determined with sufficient accuracy.

Cognitive Maps. We think that the cognitive map, as introduced by Tolman[9] on the basis of experiments with rats, are a core component of a working model of biologically plausible navigation. A cognitive map represents the environmental layout. It encodes the metrics(angles, distances) and sense relation(left versus right) between landmarks in the environment. Its representation is vectorial, and different from the 'omniscient' Cartesian-grid based maps. Research at the neural level of navigation suggest that the dead reckoning process is the key to construction of this cognitive topological map in the brain[4].

The first and major goal of this study is to find out whether coupling the visual information to proximity events leads to the development of an unsupervised landmark map which contains the basic elements of the visual and physical environment of the robocup field, using a Kohonen self-organizing map. Second, given a learned cognitive map of the environment including each learned landmark and its distances to other landmarks, the goal is to find out whether this model has been able to capture the global two-dimensional structure of the environment.

1.3 Methods

On the basis of the model requirements described above, a number of design issues have to be solved. Also, the limitations of the used robot platform have to be taken into account. First of all, a suitable visual representation is needed. Tests will be performed in a 1^{st} generation robocup field with white side boards, a blue and a yellow goal, and a green floor. As a first test, the navigation model should be able to detect the goals, the corners and other salient elements in the environment. The proximity event, which is needed to determine that the visual array in front of the robot might be a potential landmark, will be generated on the basis of the ultrasonic sensor readings. The bimodal perceptual landmark vector will consist of a reduced-resolution camera image augmented with appropriately scaled ultrasonic readings.

Image. In the experimental "Robocup" environment, important elements are distinguishable on the basis of color. For this reason, also taking into account the real-time requirements, the use of detailed shape encoding by means of, e.g., Gabor filters, has been postponed. On the basis of such design considerations, a luminance invariant portion of the well-known hue, saturation and brightness (HSB) coding, i.e., Hue-Satiation (HS) space will be used, where the original luminance of the RGB values is normalized to the maximum value. Pilot experiments revealed a desirable insensivity to lighting and weather conditions. An reduced-resolution image of 40x30 (WxH) pixels is used for the Hue and Satiation (HS) feature vector ($N_{dim} = 2400$).

Proximity. The vector of sixteen sonar sensor readings was joined with the image feature vector resulting in a bimodal perceptual input vector, a *perceptual frame* ($N_{dim} = 2416$).

Unsupervised learning for landmark detection. In order to solve the problem of automatic landmark detection and learning, the robot ran a generic collision-avoidance behavior, collecting perceptual frames whenever frontal proximity events where triggered. Considering the amount of visual elements in the environment, it was assumed that for a 5x5 Kohonen self-organizing map (SOM) it should be possible to detect up to 25 'landmarks' in the robocup environment. A 2-D Kohonen SOM was chosen because a planar neural tissue patch is a more plausible solution than its N-dimensional generalization. The advantage of the Kohonen learning scheme is that the nodes in the self-organized map will contain robust average representations corresponding to a set of similar perceptual frames in the learning history. As a result, the landmark detection will be noise tolerant. It should be noted that the topology of the SOM is not coupled to the topology of environment map in a simple way.

In this study it is assumed that salient perceptual patterns of potential landmarks can be collected on the basis of collisions or near-proximity events. Rather than expecting landmark singularities to emerge solely from the training of all visual patterns[10] which are present during all of the ego-movement, the subset of visual patterns at and around a physical collision or near proximity point are deemed essential for the development of a landmark map. Apart from the already mentioned neurophysiological support for the notion of the proximity event, it can be easily understood why multimodality is useful: It provides a strong correlation between separate modalities, concerning a single phenomenon in the environment. This may become more clear if one considers a modality which has a larger bandwidth than proximity-event detection, i.e., tactile sensing: For a looking system, a visual edge often becomes a mechanical edge at the moment tactile contact takes place.

Cognitive map representation. With the extracted perceptual representation of landmarks a cognitive or topological map of the environment will be constructed. As it is the case in animals this cognitive map will represent the relative positions of landmarks in the environment to each other. The distance between landmarks will be collected with a mechanism based on dead-reckoning (odometry). This means that at the end of an exploratory phase, a matrix with distances Δ_{ij} between a landmark i and a landmark j will exist as a representation within the navigation system.

Robot platform and Environment. The robot used for this research experiment is the Pioneer II DX robot. The physical characteristics of the robot are length 44 cm, width 33 cm and height of the body 22 cm. This robot is equipped with an array of 16 sonar transducers that provide range information of objects in the environment from 10 cm to more than five meters. On top of the robot a color CCD camera is mounted with a resolution of approximately 440k pixels. An optical revolution counter on the

axles, measuring 19 ticks per traveled millimeter is used for speed sensing and dead-reckoning during linear trajectory segments only. As regards the software platform, the different tasks and behaviors used on the robot in these experiments were implemented on our modular agent-based architecture 'XSAM'. The environment used in conducting the experiments was the *robocup field* in the robotics lab of the AI Institute at Groningen University. This rectangular green field consist of white boarding with a height of approximately 40 cm on each side. The dimension of the field is approximately 4 by 4.60 meters. On each of the two short sides of the field is an colored area that represents a goal. The colors of these goals are yellow and blue. In the robotics lab the lighting is provided by fluorescent ("TL") illumination and a window positioned at the side of the yellow goal. When standing in front of the blue goal there is a darker side with a wall on the right hand and a brighter side with computer terminals on the left hand.

2 Results

After a number of pilot experiments, it was determined that a driving speed of 0.2 m/s, a frame rate of 2 Hz and a proximity threshold of 1 m yielded a useful 'proximity set', i.e., the subset of all perceptual frames consisting of those frames which are collected at proximity events. Experimental trials are of the order of 20-30 min., collecting 2400-3600 frames, of which 500-700 perceptual frames are classified as proximity events. The Kohonen training parameters were: A 5x5 network, using 200 epochs, starting with a radius of 100% and a learning rate of 0.99 and ending with a radius of 0% and a learning rate of 0.01. A steep and non-linear parameter curve for radius and learning rate was used in order to prolongate the relative duration of the final fine-tuning stage. At the end of training, the root-mean square (rms) error between the SOM and the N training vectors can be computed: $\epsilon_{rms} = ((\sum_i^N \sum_j^F (x_{ij} - m_{kj})^2)/NF)^{0.5}$, where $i = [1, N]$ is the sample index, $j = [1, F]$ is the feature-value index and k is the index of the nearest neighbor cell in the SOM, i.e., m_k, to the input sample x_i. The rms error value for four networks, trained on four days with different weather and light conditions varied from 25-41 and there were no appreciable differences between same-day and different-day cross comparisons, using the HS images. As expected, rms errors were larger for an RGB image representation, using the same scaling (0-255). Figure 1 shows the resulting SOM, or rather: Kohonen Landmark Map (KLM), after training on HS images. The images on the right represent the sets of raw feature frames which share the same nearest neighbor in the KLM.

In the visual Kohonen Landmark Map (Figure 1) patterns of important landmarks in the environment such as the blue and yellow robocup goal, which are easily perceptible and unique, clearly emerge (in grey-scale print, the blue goal looks dark). The raw feature frames which have one of these landmark representations(blue and yellow goal) as the same nearest neighbor in the KLM show a clear homogeneity. Landmarks representing points in space which are not easily perceptible or unique on the visual pattern show more heterogeneity in the set of feature frames with the same nearest neighbor. For example corners and the points on each side of the field reveal great similarity in the visual domain, whereas the effective use of the sonar information may help to disambiguate[3].

At this point, it is an important question whether the amount of information collected by the system thus far is sufficient to construct an approximate map of the 2-D environment. Given a set of N points in the 2-D plane, an NxN distance matrix can be constructed. It can be easily shown that it is possible to estimate the position of a point i in this plane on the basis of a linear transform on its distances to the other $N-1$ points,

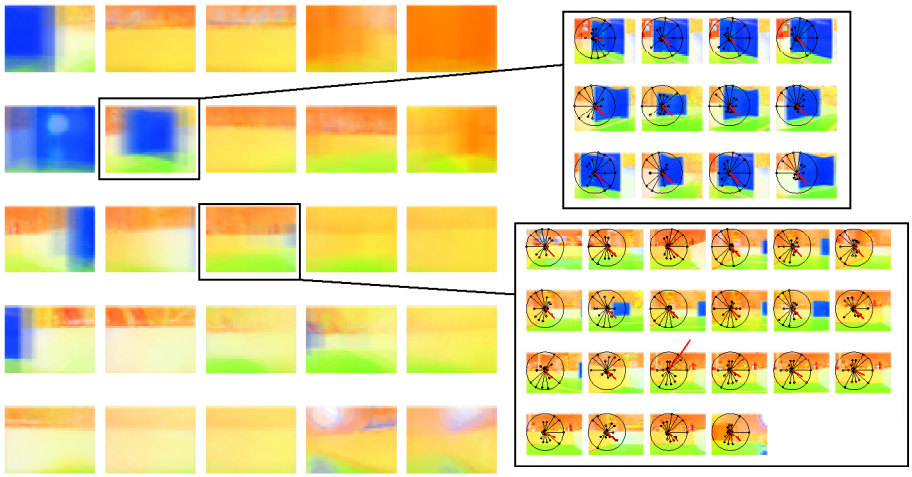


Figure 1: Left : KLM after training on HS images; Right : sets of raw feature frames belonging to node (2,2) and node (3,3) respectively as the nearest neighbors in the KLM. The main image of the blue goal is at row 2, column 1. The main image of the yellow goal is at row 1, column 5 (1-based indexing is used here). The black circle and arrows in each cell represent the average sonar data as $1/r$ (long='near') belonging to the landmark image. It can be seen that the image data dominate the sonar patterns.

by solving a set of linear equations and using known coordinates as the desired output:

$$\hat{x}_i = \beta_0 + \beta_1 \Delta_{1i} + \beta_2 \Delta_{2i} + ... + \beta_N \Delta_{Ni} \quad (1)$$

$$\hat{y}_i = \gamma_0 + \gamma_1 \Delta_{1i} + \gamma_2 \Delta_{2i} + ... + \gamma_N \Delta_{Ni} \quad (2)$$

where β and γ refer to the linear coefficients for x and y, respectively.

In other words, according to this model, the distance matrix Δ contains information on the dimensionality and geometry of the space containing the N points. By determining the Eigenvectors of Δ, an unsupervised estimate of the 2-D structure can be obtained theoretically, on the basis of the two axes with the largest Eigenvalues, using principal-components analysis (PCA). The resulting representation will need to be normalized on the basis of an affine transform (4 parameters).

To illustrate this principle, an artificial distance matrix was constructed which represented the distances between 25 points uniformly distributed in a 5 by 5 square. One percent of the distances was not realized and replaced by a random value within the same range as the distance values. On all distances, uniform noise with a maximum of 0.3 was added. Figure 2 shows the grid points and the scaled estimates which are based on the PCA method. As can be seen, the reconstruction is not without error, especially when considering points at the perimeter of the field, but the general layout is reconstructed. Noise on the distances in the matrix will increase the error in a reconstruction of the map somewhat. Missing distances have a stronger bias effect. This result on artificial data illustrates, that a landmark-exploring agent can construct an approximate vectorial map of a limited environment, in principle.

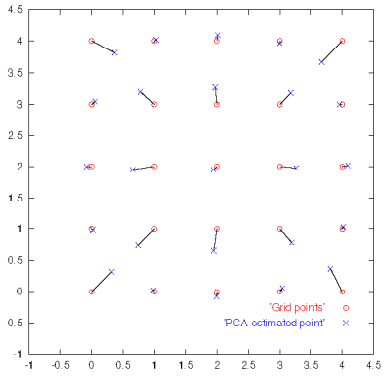


Figure 2: PCA results on artificial created distance matrix between 25 points, with 1% missing links and a white noise of max. 0.3 on the distances, in this scale. Circles mark the original 2-D grid points, the x crosses represent the estimated grid points according to the two largest Eigenvectors of the distance matrix.

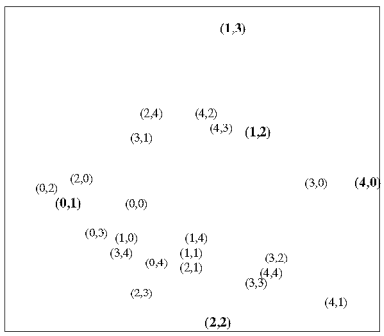


Figure 3: Topological landmark representation based on PCA of the distance matrix. Without human intervention, the system has correctly place the blue(0,1) and yellow(4,0) goals in opposing positions. The landmark views on the room's window, i.e. nodes [(1,2),(1,3)] at the top, indicate that the y-axis should be flipped in order to obtain the real-world configuration.

In order to test this model on real data, the robot ran through the environment, using a given 5x5 KLM and noting both the identities of each landmark image and its dead-reckoned distance to the preceding landmark. From the resulting data a 25x25 distance matrix was filled. Distances for landmark transitions which did not occur during the test run were replaced random noise with mean and standard deviation of the known distance distribution. Figure 3 show the topological landmark representation of the environment which resulted after PCA on the distance matrix. The labels in the topological map correspond to the labels in the perceptual landmark of the KLM used. The map represents the two main axis of the robocup field: Blue opposite to the yellow goal and 'terminals' side opposite to the 'dark wall' side.

3 Conclusion

We have proposed a biologically inspired landmark-exploration model, using multimodality and unsupervised learning. Furthermore, it was illustrated that a landmark-distance matrix - such as may be determined on the basis of dead reckoning - contains the necessary information for the development of a vectorial 2-D map. By using principal-component analysis, the two major eigenvectors were illustrated to refer to the major Cartesian axes of an environment, both for artificial grid data and for real exploratory navigation on a robocup field. However, it should be noted that more research is needed, since the model is limited in a number of ways. The fixed dimensionality of a Kohonen map does not seem to be biologically plausible. Furthermore, a calibration of the PCA-based cognitive map is needed, requiring an affine transform (4 parameters) in order to scale and rotate the internal map appropriately relative to motion control. In principle, also this last step may be unsupervised, if tuning behavior is added to the navigation system. **Future research** will be directed at an extension of the model, using behavioral strategies. For example, at a proximity event and a detected landmark, looking left and right will allow for a more unique coding of a landmark, coupling it more strictly to the geographical location in the real world, but also providing for relative angular information which couples the landmark to its neighbors during the common exploration behavior.

References

[1] B.A. Cartwright and T.S. Collett. Landmark learning in bees. *Journal of Comparative Physiology*, 151:521–543, 1983.

[2] Colby C.L. and M.E. Goldberg. Space and attention in parietal cortex. *Annual Review of Neuroscience*, 22:319–349, 1999.

[3] R. Fehrmann. Cognitive navigation modeling in robots. Technical Report [MSc Thesis], AI Inst./Rijksuniversiteit Groningen, 2002.

[4] B.L. McNaughton, C.A. Barnes, J.L. Gerrard, K. Gothard, M.W. Jung, J.J. Knierim, H. Kudrimoti, Y Qin, W.E. Skaggs, M. Suster, and K.L. Weaver. Deciphering the hippocampal polyglot: The hippocampus as path integration system. *Journal of Experimental Biology*, 199:173–185, 1996.

[5] R.U. Muller and J.L. Kubie. The effects of changes in the environment on the spatial firing of hippocampal complex-spike cells. *Journal of Neuroscience*, 7:1935–1950, 1987.

[6] J. O'Keefe and J. Dostrovsky. The hippocamplus as a spatial map: Preliminary evidence from unit activity in the freely moving rat. *Brain Research*, 34:171–175, 1971.

[7] Wehner R. and F. Raber. Visual spatial memory in desert ants. *Cataglyphis bicolor*, Experientia 35, 1979.

[8] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proc. IEEE Conf. on Robotics and Automation ICRA*, San Francisco, CA, 2000. IEEE.

[9] E.C. Tolman. Cognitive maps in rats and man. *Psychological Review*, 55:189–208, 1948.

[10] Georg Von Wichert. Selforganizing visual perception for mobile robot navigation. In *1st EUROMICRO Workshop on Advanced Mobile Robots*, 1996.

Advanced information search within a research-based multinational

Sander Spek [a] Kees-Jan van Dorp [a] Etienne Mathijsen [b]
Theo de Haas [b] Jaap van den Herik [a]

[a]Universiteit Maastricht, IKAT, P.O. Box 616, 6200 MD Maastricht, The Netherlands
{s.spek,c.vandorp,herik}@cs.unimaas.nl
[b]DSM N.V., Corporate ICT (I-Net), P.O. Box 601, 6160 AP Geleen, The Netherlands
{etienne.mathijsen,theo.haas-de}@dsm.com

Abstract

The paper deals with advanced information search within a Dutch, knowledge-intensive multinational. Seven challenges to improve the company's information search are identified. These challenges are matched with advanced information search approaches from the literature, ranging from statistical analysis to culture change. Special attention has been given to automatic categorization and personalization; solutions that will lead to considerable better queries. It even reduces the information inconsistency which is a well-known obstacle in the company's infrastructure.

1 Introduction

In the last decade, our modern world has experienced a - sometimes fast, sometimes bumpy - transition from an industrial-based economy to a knowledge-based economy. The transition is now nearly complete[1] and irreversible [11]. This development has enriched the literature with new many concepts, such as knowledge management, intellectual capital, learning organizations, the organizational memory, and electronic document management (e.g., [1, 10, 11]). Digital libraries are a further advanced example of these concepts. Several issues of the digital libraries have been investigated, amongst others user behaviour [7, 8], relevance from a philosophical standpoint [13], and many technical issues such as suitable architectures [3].

The article aims at contributing to knowledge management and information systems research by analysing the empirical company challenges of advanced information search. It focuses on the question which challenges the company must take up so as to improve the efficiency and effectiveness of information search in digital libraries. The notion of advanced information search is leading in our problem statement: *Which advanced information search approaches are identified in the literature that have the potential to fulfil the empirical needs of a knowledge-intensive multinational?*

The research is performed in the digital library of a Dutch, research-based and knowledge-intensive multinational producing life science products, performance

[1]Meaning that we are acquainted with its concepts, although they are far from solidified.

materials, and chemicals. The method consists of interviews with employees, knowledge managers, and ICT experts, all working at the company. The research is limited to the search engine of the company's intranet, and to a digital library system for research articles.

The knowledge produced and stored within the organization's systems is voluminous and has a complex nature. The organization itself is divided into multiple groups, each having their own domain of expertise. Facilitating co-operation and knowledge-exchange between these different groups is a research challenge per se.

Several empirical challenges for the company are extracted in section 2. Advanced approaches from the literature are examined on their usefulness to help solving the challenges in section 3. Section 4 matches the approaches with the challenges. Finally, section 5 elaborates on the application of the two most interesting approaches.

2 Challenges

The company under investigation is a large and knowledge-intensive organization. Even though the information search system functions satisfactorily, closer inspections show that nevertheless improvements are needed. Below, we list seven challenges that are identified within the company.

2.1 Query formulation

Since the way users do formulate queries is rather imperfect for search engines one challenge is to improve query formulation. The current literature supports this idea and mentions four topics to be improved upon (adapted from [4, 17]):

- *One-term queries:* Most users formulate their query using only one word or one term. Better results are obtained by using multiple relevant terms, combined with relational statements like 'and', 'or', 'not', and 'near'.
- *Bad constituents:* Some constituents are not suitable for use in search queries. Examples are stopwords, verbs, and prepositions.
- *Bad terms:* Commonly used words and ambiguous words should be avoided since they would result in too many documents. In multiple-term constructions 'and' or 'near' can be used.
- *Misspelled words:* Users generally are not perfect spellers and sometimes they also type too hasty.

Imperfect queries generally result in not meeting the user's information need. Then there is a risk that the user becomes frustrated by the lack of results, and classifies the system as being 'bad' [7]. Some search engines remove bad constituents and suggest spelling corrections.

2.2 Information overflow

Nowadays information is available in large amounts. The intranet of the company consists of over 70 websites owned by the different departments and groups. The digital library application on the date of submission contains some 3.5 million documents. The challenge is to avoid document overflow after a query that yields over hundreds of documents, e.g., by additional filtering.

2.3 Document formats

The retrieval of information from web-based collections relies on mainly HTML-documents. However, owing to the use of multimedia, many documents are enhanced with other document formats (e.g. images, flash animations) and some documents are not in HTML-format at all, like PDF-files.[2] Moreover, some web accessible information - or rather: data - is 'hidden' inside databases, and is only accessible by means of web forms.

The challenge is to cope with these different formats, for instance by issuing a standardized workplace so that all users have the same tools, or by applying converters.

2.4 Information inconsistency

Information is often not formatted in a consistent manner. For instance, every business group is advised to have a page with employee information. However, since it is not obligatory and names are not prescribed, department A has as *Staff* page, department B has an *Employees* page, department C has a *Who Is Who* page, and, on top of that, department D does not have a page like this at all.

One respondent stated that for the business groups themselves this is not a problem. They can find everything at their own intranet site. However, on the site of another group they will be compeltely lost.

The challenge for effective knowledge sharing is to have a company-wide information structure and terminology, either by policy or by tools. However, reducing the webmaster's freedom will also reduce his/her job satisfaction.

2.5 Meta-data

When users publish information they are not inclined to spend valuable time adding decent meta-data to their information. Most users consider their name and department enough information, and do not realize how valuable keywords and summaries can be to the retrieval of their document.

It is a challenge to have sufficient and consistent meta-data added to documents. At the company's digital library this is handled by obtaining meta-data from specialized suppliers. Even though the quality of this meta-data is high, inconsistencies between the different suppliers can still trouble the search process.

2.6 Integration

Knowledge sharing within the investigated organization is supported by multiple software products. Products exist for web meetings, project management, document management, intranet and internet publishing, and a search engine for the intranet and internet. In the ideal situation, one would want these applications to be seamlessly integrated. For example, final versions of documents in the project management application will be copied to the document management system, the search engine can search the document management system respecting the searchers access rights, etcetera. In practice, this is a troublesome challenge.

[2] Even when the search engine is capable of indexing all sorts of exotic document formats, for the information need to be satisfied the user still needs the right tool to view the document.

Especially without the help of the producers of the software this is almost impossible.

2.7 Presentation

The common presentation of information retrieval results is very primitive and looks like in the past years no progress at all has been made in this field. Without exception, all search engines return their results in a long list, ranked at relevancy. To the user this is not very appealing, and it is hard to find the relevant documents from the list. Here, the challenge lies in finding a presentation that better suits the user's behaviour.

3 Approaches

In this section, approaches from the literature are examined with the aim to improve the information search facilities. We will discuss the seven most promising ones. These are automatic categorization, conceptualisation, culture change, information integration, application integration, personalization, and statistical analysis. Special attention has been given to the question whether these challenges are more benificial within the boundaries of a multinational instead of the public Internet.

3.1 Automatic Categorization

Automatic categorization [5, 14, 18] can be applied in conjunction with a search engine. The first possibility is the way Google and Yahoo have implemented it: to provide the user with a classification and the facility to search within a certain branch of that classification. Also it can be used to narrow down ambiguous queries. When a term appears often in two different branches of a taxonomy, probably the same term refers to different concepts. The user can then be asked to select the desired one. Automatic categorization can also be used to improve the presentation of large amounts of returned results. Instead of showing everything in one long ranked list, several groups of related results can be presented [22].

Taxonomies are more easy to produce in a bounded domain. However for a knowledge intensive multinational of the size of the company in question, this still is a tremendous effort. And also it is still subject to continuous changes.

3.2 Conceptualisation

A conceptualisation is 'the collection of objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them' (Free On-line Dictionary of Computing, `http://www.foldoc.org`). These concepts can subsequently be used as the basis for classification (the categories), query recommender systems (suggesting the user to add a concept to the query) or for visualization techniques [19, 20].

Since conceptualisation also makes use of taxonomies, the same remarks as for automatic categorization (par. 3.1) apply here.

3.3 Culture change

Not everything can be solved by using the most advanced techniques and tools. The quality of an information system is for a great part determined by humans

using the system. To achieve optimal performance with an information system, the user culture has to be taken into account. There are several ways to introduce a culture change, from policy making to educational courses. For instance, people can be taught to compose multiple keyword queries, or can be made aware of the importance of correct meta-data.

Culture change is more easy to be accomplished when learning facilities and information exchange mechanisms are already present. It requires a good social infrastructure.

3.4 Information integration

In order to get the message through, information must be integrated. This means both a common understanding of concepts and a common terminology, as well as a common structure and layout. For instance, think back at the problems we saw in section 2.4 with the employee information pages. Or think of researchers talking about acetylsalicylic acids, to a sales manager who is only familiar with it's brand name: aspirin.

Information integration is an effort that is impossible to succeed on the large scale of the Internet. Certainly not without the help of semantic tools like WordNet. On a company scale however, this could provide an adequate solution.

3.5 Application integration

Tools used in an organization often do not all come from the same package or vendor (like Microsoft Office containing word processing, spreadsheets, presentation tools, etcetera, and Microsoft operating systems). Integration problems as discussed in section 2.6 will be eminently present. It is important for the various applications to be able to share information. Solutions may be found in customizable information schemes like XML, or other shared communication standards.

Application integration is easier to acchieve on a company scale since the number of applications used is limited, and often good contact with the vendors exist.

3.6 Personalization

Personalization, or customization, allows users to obtain results that match their personal interest, represented by a profile [2, 9, 16]. Imagine two people searching for information using the following query: *church Saint Servais Maastricht*. Obviously, they want information on the church of Saint Servais in the city of Maastricht. With this query, we cannot distinguish between the information needs of these people. But suppose that we know that the first person recently visited a website for booking hotels in the city of Maastricht and the website of the Maastricht tourist agency. And the other person is searching from an IP address belonging to the history department of a university. At this point it is easy to pinpoint the differences in information need: the first person wants touristic information on the church, while the second one is a researcher interested in its history.

The field of personalization falls into three categories: rule-based, content-based, and collaborative filtering (e.g., [6, 15]). Rule-based personalization uses simple rules to compose a profile. Content-based filtering examines documents one has taken an interest in previously (e.g., the web browsing history) and constructs

a profile out of that. By using relevance feedback this profile can be improved more than by simply studying the history, but this will require more user effort [2, 16]. Collaborative filtering on the other hand compares one user to similar users (e.g., colleagues with the same research interest, or complete strangers with a similar browsing history) and composes the profile out of their preferences.

3.7 Statistical Analysis

In the field of knowledge management statistical analysis research applies mostly to data- and webmining and related techniques [12]. Datamining is the automated or semi-automated process of pattern discovery in electronic data [21]. Because of this automation, datamining is suited to analyse large amounts of data, and supply patterns to get new insights into the data. Webmining applies this to web documents.

In information retrieval applications, statistical analysis can be used to find relations between terms and documents, and to calculate the importance of certain documents (e.g., number of downloads, or number of references to a document). This can be used to provide the user with additional search terms, additional relevant documents and a better estimation of the importance of a document, enabling a smaller result list by chucking out the irrelevant ones.

Statistical analysis is one solution that is harder to achieve on a company scale. Especially for datamining large data collections are preferred.

4 The matching of challenges and approaches

The challenges from section 2 can be dealt with by one or more of the approaches discussed in section 3. Therefore, a match between the challenges and the approaches can be made. This is done in table 1. It shows the challenges on the left side, and approaches on the top row. The +-signs show which approaches are suitable for each challenge.

5 Applications

Below we will briefly discuss the fields of automatic categorization and personalization. These are the topics that are selected for a deeper investigation within the company framework. In particular, the latter is prompted by business interests.

5.1 Automatic categorization

Automatic categorization improves query formulation by allowing the user to limit the search to certain categories. Also ambiguous queries can be narrowed down. The addition of meta-information profits from uniform, company-wide taxonomy and classification rules. It can improve the information inconsistency by identifying global concepts. Finally, the presentation of search results can be improved by automatically categorizing them.

5.2 Personalization

Personalization provides three significant improvements related to the challenges discussed. First, it creates better queries by adding information that the user would omit unintentionally (e.g., the user's location, the user's interest). Second,

	categorization	conceptualisation	culture change	information integration
query formulation	+	+	+	+
information overflow				
document formats			+	
information inconsistency	+		+	+
meta-data	+		+	+
integration			+	+
presentation	+	+		

	application integration	personalization	statistical analysis
query formulation		+	+
information overflow		+	+
document formats	+		
information inconsistency	+	+	
meta-data			
integration	+		
presentation	+		+

Table 1: Matching challenges and approaches.
A + indicates that the approach (top row) can be used to tackle the challenge (left column).

large amounts of results can be filtered using a profile. In this way the problem of information overflow will be reduced. Finally, problems with information inconsistency will reduce, too. Similar users can use different terms and still find each others information, because their profiles match by other terms.

6 Conclusions

The aim of the research was to identify approaches from the literature that can be applied to solve empirical challenges for a knowledge-intensive multinational. The discussed approaches are automatic categorization, conceptualisation, culture change, information integration, application integration, personalization, and statistical analysis. These approaches are mapped to the challenges they can help solving. This is presented in table 1.

Based on this research and on business interests, the decision has been made to investigate the possibilities and effects of automatic categorization and personalization on the digital library environment within the company.

References

[1] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek. Towards a technology for organizational memories. *IEEE Intelligent Systems*, pages 40–48, May/June 1998.

[2] W. B. Croft, S. Cronen-Townsend, and V. Lavrenko. Relevance feedback and personalization: A language modeling perspective. In *Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, June 2001.

[3] L. Feng, M. A. Jeusfeld, and J. Hoppenbrouwers. Towards knowledge-based digital libraries. *SIGMOD Record*, 30(1):41–46, March 2001.

[4] N. L. Fielden and N. Kuntz. *Search Engines Handbook*. McFarland, 2002. 0-7864-1308-5.

[5] G. W. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization of the web and identification of communities. *IEEE Computer*, 35(3):66–71, 2002.

[6] IBM. *Web Site Personalization*, February 2000. Technical report by the High Volume Web Site Team.

[7] F. F. Jacobson and E. N. Ignacia. Teaching reflection: Information seeking and evaluation in a digital library environment. *Library Trends*, 45(4):771–802, Spring 1997.

[8] S. Jones, S. J. Cunningham, and R. J. McNab. An analysis of usage of a digital library. In *European Conference on Digital Libraries*, pages 261–277, 1998.

[9] S. Lawrence. Context in web search. *IEEE Data Engineering Bulletin*, 23(3):25–32, 2000.

[10] B. Lemken, H. Kahler, and M. Rittenbruch. Systained knowledge management by organizational culture. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

[11] D. Logan and C. Young. Challenging the assumptions of the knowledge economy. Gartner Research Document, October 2001. AV-14-7280.

[12] F. Menczer. Complementing search engines with online web mining agents. *Decision Support Systems*, 35:195–212, 2003.

[13] S. Mizzaro. Relevance: the whole history. *Journal of the American Society for Information Science*, 48(9):810–832, 1997.

[14] H.-L. Ong, A.-H. Tan, J. Ng, H. Pan, and Q.-X. Li. Organizing and personalizing intelligence gathering from the web. *Internation Journal of Intelligence Systems in Accounting, Finance and Management*, 11(1):9–21, 2002. Special issue on Knowledge Management.

[15] N. Ramakrishnan. Pipe: Web personalization by partial evaluation. *IEEE Internet Computing*, pages 21–31, November/December 2000.

[16] S. Shearin and H. Lieberman. Intelligent profiling by example. In *International Conference on Intelligent User Interfaces (IUI)*, January 2001.

[17] C. Sherman and G. Price. *The Invisible Web: Uncovering Information Sources Search Engines Can't See*. CyberAge Books, 2001. 0-910965-51-X.

[18] K. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[19] A. Veling. The aqua browser - visualisation of dynamic concept spaces. *Journal of the Association for Global Strategic Information*, 6(3):136–142, November 1997.

[20] F. Wiesman. *Information Retrieval by Graphically Browsing Meta-Information*. PhD thesis, Universiteit Maastricht, 1998.

[21] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan-Kaufmann, 1999.

[22] Y.-F. B. Wu, C. Rakthin, and C. Li. Summarizing search results with automatic tables of contents. In *Eighth Americas Conference on Information Systems*, August 2002.

Online Adaptation of Computer Game Opponent AI

Pieter Spronck Ida Sprinkhuizen-Kuyper Eric Postma

Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht

Abstract

Online learning in commercial computer games allows computer-controlled opponents to adapt to human player tactics. For online learning to work in practice, it must be fast, effective, robust, and efficient. This paper proposes a technique called "dynamic scripting" that meets these requirements. In dynamic scripting an adaptive rule-base is used for the generation of intelligent opponents on the fly. The adaptive performance of dynamic scripting is evaluated in an experiment in which the adaptive players are pitted against a collective of manually designed tactics in a simulated computer roleplaying game. The results indicate that dynamic scripting succeeds in endowing characters with adaptive performance. We therefore conclude that dynamic scripting can be successfully applied to the online adaptation of computer game opponents.

1 Introduction

The quality of commercial computer games is directly related to their entertainment value [9]. The general dissatisfaction of game players with the current level of artificial intelligence for controlling opponents (so-called "opponent AI") leads to their preference for human-controlled opponents [6]. Improving the level of opponent AI (while preserving the characteristics associated with the entertainment value [7]) is desired in case human-controlled opponents are not available or not feasible.

For complex games most game AI developers resort to scripts [10]. Controlling opponents with complex abilities requires these scripts to be quite long [1]. Two major weaknesses of long scripts are that (1) they are prone to containing errors because they are complex, and (2) they cannot deal with unforeseen player tactics because they are static. As a result, a human player can easily defeat supposedly tough opponents by exploiting the "holes" in their scripts. Evidently, easily defeatable opponents hamper the entertainment value of commercial computer games.

In our view, there are two ways to improve the quality of scripted opponent AI. The first way is to employ offline learning prior to the release of a game to deal with the problem of holes in the scripts that control the opponents. Our earlier work showed offline learning techniques to be successful in the identification of holes and even in the discovery of novel tactics [8]. The second way of improving the quality of scripted opponent AI is to apply online learning after the game has been released. Online learning allows the opponents to adapt to changes in human player tactics. Even though it has been shown to be feasible for simple games [2], unsupervised online learning is widely disregarded by commercial game developers [11] and literature on unsupervised online learning in commercial computer games is scarce. However, we believe it to be

of great potential for improving the entertainment value of commercial computer games.

Our research question reads: How can unsupervised online learning be incorporated in commercial computer games? This paper proposes a novel technique called "dynamic scripting", that realises unsupervised online adaptation of scripted opponents, and reports on preliminary experiments performed to assess the adaptive performance obtained with dynamic scripting.

The outline of the remainder of the paper is as follows. Section 2 discusses AI in computer roleplaying games. Section 3 describes our dynamic scripting technique. The experimental setup for evaluating the adaptive performance of dynamic scripting is described in section 4. The results of the experiments are presented and discussed in sections 5 and 6, respectively. Finally, section 7 concludes that dynamic scripting has the potential to be successfully incorporated in commercial games.

2 Computer Roleplaying Game AI

In Computer RolePlaying Games (CRPGs) the human player is situated in a virtual world represented by a single character or group of characters called a "party". Each character is of a specific type (e.g., a fighter or a wizard) and has certain characteristics (e.g., weak but smart). In most CRPGs, the human player goes on a quest, which involves conversing with the world's inhabitants, solving puzzles, discovering secrets, and defeating opponents in combat. During the quest the human-controlled characters gather "experience", thereby gaining more and better abilities, such as advanced spell-casting powers. Some examples of modern CRPGs are BALDUR'S GATE, NEVERWINTER NIGHTS, MORROWIND and EVERQUEST.

Opponent AI in CRPGs is almost exclusively based on scripts, i.e., lists of rules that are executed sequentially. Scripts have four main advantages; they are (1) understandable, (2) easy to implement, (3) easily extendable, and (4) useable by nonprogrammers [10]. Usually scripts are implemented in a formal language that has functions to test environmental conditions, to check a character's status, and to express commands. During the game-development phase scripts are manually adapted to ensure they exhibit the desired behaviour. After the game is released the scripts and associated behaviours remain unchanged (unless game patches are issued to update the scripts).

To deal with all possible choices and all possible consequences of actions the scripts controlling the opponents are of relatively high complexity. In contrast to classic CRPGs, such as the ULTIMA series, in modern CRPGs the human and opponent parties are often of similar composition (see figure 1 for an example), which entails that the opponent AI should be able to deal with the same kind of complexities the human player faces. The challenge such an encounter offers can be highly enjoyable for human players.

Figure 1: An encounter between two parties in BALDUR'S GATE.

However, the scripts controlling the opponents cannot anticipate on all tactics exhibited by human players. As a consequence, human players usually have little trouble in identifying and exploiting the weaknesses in the opponents. Since these weaknesses usually permeate throughout the entire game, this eventually leads to a decrease in entertainment value of the game.

3 Dynamic Scripting

We introduce dynamic scripting as a technique to overcome the limitations of static scripts in CRPGs. In dynamic scripting, the scripts controlling the opponents are modified during the game to adapt to the tactics of the human player. For online learning to work in practice, it must be fast (computationally cheap), effective (maintain at least the quality of the unadapted scripts), robust (able to deal with the inherent randomness of computer games), and efficient (require few experiments).

To achieve the goal of fast, effective, robust and efficient dynamic scripting, we need a learning algorithm of high performance. The two main factors of importance when attempting to achieve high performance for a learning mechanism are using deterministic experiments and adding knowledge [4]. The nature of our environment precludes determinism, so in our case it is imperative that the learning process is based on knowledge. To this end, our dynamic scripting technique relies on a rulebase that contains manually designed rules based on domain-specific knowledge.

The dynamic-scripting process is illustrated in figure 2. A single rulebase is associated with every opponent. Each rule in the rulebase has a weight indicating its importance. At the start of an encounter, a new script is generated for each opponent by randomly selecting a fixed number of rules from its rulebase. The probability of a rule being selected depends on its weight; i.e., rules with larger weights have a higher probability of being selected.

The learning mechanism in our dynamic scripting technique is inspired by reinforcement learning techniques [5]. It has been adapted for use in games because

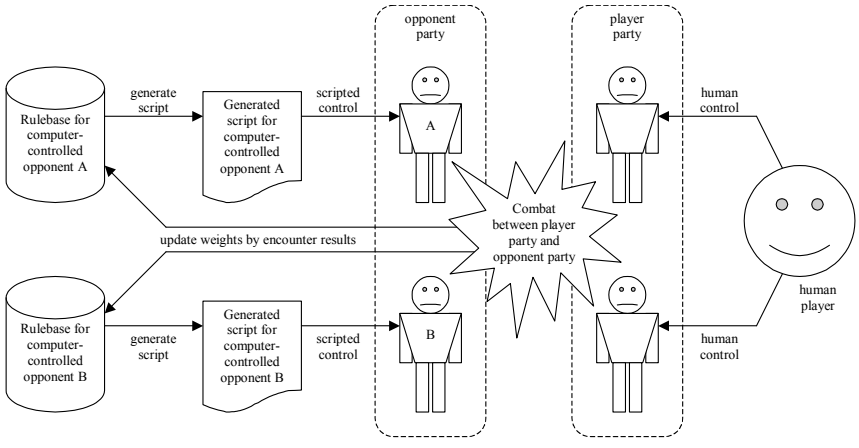


Figure 2: The dynamic-scripting process. For each computer-controlled opponent a rulebase generates a new script at the start of an encounter. After an encounter is over, the weights in the rulebase are adapted to reflect the results of the fight.

regular reinforcement learning techniques are unsuitable for this purpose, in particular since (1) it is difficult to decide what information should be placed in a state vector, and (2) reinforcement learning generally adapts too slowly for online use in games [3]. Our implementation is as follows. Upon completion of the encounter, the weights of the rules employed in the combat are adapted depending on their contribution to the outcome. Increasing the weight rewards rules that lead to success, whereas decreasing the weight punishes rules that lead to failure. The remaining rules get updated so that the total weight of the rules in the rulebase remains unchanged.

The dynamic-scripting technique meets at least three of the four requirements listed above. First, it is fast because it only requires the extraction of rules from a rulebase and the updating of weights once per encounter. Second, it is effective because all rules in the rulebase are based on domain knowledge (although they may be inappropriate for certain situations). Third, it is robust because rules are not removed immediately when punished. The dynamic-scripting technique is believed to meet the fourth requirement of efficiency because with appropriate weight-updating parameters it can adapt already after a few encounters. To determine whether this belief is warranted, we performed an experiment with dynamic scripting in a simulated CRPG situation.

4 Experimental Setup

This section describes the experimental setup used to test the efficiency of dynamic scripting in CRPGs. It describes the problem situation to which dynamic scripting is applied (4.1), the scripts and rulebases (4.2), the fitness functions (4.3), the learning parameters (4.4) and the actual experiments (4.5).

4.1 The CRPG Simulation

The gameplay mechanism in our CRPG simulation was designed to resemble the popular BALDUR'S GATE games (see figure 1). These games contain the most complex and extensive gameplay system found in modern CRPGs, closely resembling classic "pen 'n paper" roleplaying games. Our simulation entails an encounter between player and opponent parties of similar composition. Each party consists of two fighters and two wizards of equal experience level. The armament and weaponry of the party is static; each character is allowed to select two (out of three possible) magic potions; and the wizards are allowed to memorise seven (out of 21 possible) spells. The spells incorporated in the simulation are of varying types, amongst which damaging spells, blessings, curses, charms, area-effect spells and the summoning of allies.

The choices for potions and spells are made before the encounter starts and depend on the (generated) scripts. In the simulation, this is done as follows. Before the encounter starts the script is scanned to find rules containing actions that refer to drinking potions or casting spells. When such a rule is found, a potion or spell which can be used in that action is selected. If the character controlled by the script is allowed to possess the potion or spell, it is added to the character's inventory.

4.2 Scripts and Rulebases

The scripting language is designed as to define rules composed of an optional conditional statement and a single action. The conditional statement consists of one or

more conditions combined with logical ANDs and ORs. Conditions can refer to a variety of environmental conditions, such as the distances separating characters, the characters' health, and the spells that are suffered or benefited from. There are five basic actions: (1) attacking an enemy, (2) drinking a potion, (3) casting a spell, (4) moving, and (5) passing. In the scripting language, spells, potions, locations and characters can be referred to specifically (e.g., "cast spell 'magic missile' at closest enemy wizard"), generally (e.g., "cast any offensive spell at a random enemy") or somewhere in-between (e.g., "cast the strongest damaging spell available at the weakest enemy").

Scripts are executed in sequential order. For each rule the condition (if present) is checked. If the condition is fulfilled (or absent), the action is executed if it is both possible and useful in the situation at hand. If no action is selected when the final rule is checked, the default action 'pass' is used.

As explained in section 3, the rulebase consists of a list of weighted rules. The weight determines the probability that a rule from the rulebase is selected for the script that is generated at the start of an encounter. In addition, each rule is assigned a priority. The priority determines the position of the rule in the script. If two rules with the same priority are selected, the rule with the highest weight value gets precedence over the other. If both rules have the same weight value, their ranking is determined randomly.

4.3 The Fitness Functions

For the weight adaptation mechanism two fitness functions are used in the CRPG simulation: a fitness function for the party as a whole, and a fitness function for each individual character. The fitness of a party is a value on the unit interval [0,1]. The fitness is defined to be zero if the party has lost the fight, and 0.5 + half the average remaining health of all party members if the party has won the fight.

The fitness of a character is also a value on the unit interval [0,1], that is based on four factors, namely (1) the average remaining health of all party members (including the character), (2) the average damage done to the opposing party, (3) the remaining health of the character (or, if he died, the time of death) and (4) the party fitness. The fitness function for individual characters assigns a large reward to a victory of its party (even if the individual itself did not survive), a smaller reward to the individual's own survival, and an even smaller reward to the survival of its comrade party members and the damage they inflicted on the opposing party. This definition therefore attempts to illicit the emergence of successful party behaviour, and in a lesser sense the aim of a character to ensure its own survival.

4.4 The Learning Parameters

The learning parameters determine how the character fitness of a script is translated into adaptations of the weights associated with the rules in the rulebase. For our experiment we set the break-even point to 0.3, i.e., with a character fitness lower than 0.3 the weights of the rules executed during the encounter were penalised whereas those with a fitness higher than 0.3 were rewarded. All weights in the rulebase were initialised with a value of 100 and were constrained to values between 0 and 2000.

The maximum reward (awarded to scripts with the maximum fitness of 1) was set to 100. The maximum penalty (issued for a minimum fitness of 0) was set to 30. At the break-even point the weights are not adapted. For other fitness values the weight

adaptation is calculated as a linear mapping between the break-even point and the maximum, e.g. a fitness of 0.65 was rewarded with 50 points. To keep the sum of all weight values in a rulebase constant, weight changes correspond to a redistribution of weights in the rulebase. Note that the weight shifts in the rulebase will, if successful, generate opponent behaviour that favours winning a fight. Whether or not this is fun for the human player is currently not taken into account, although stronger computer-controlled opponents are usually considered to be more fun for experienced players.

The size of the script for a fighter was set to five rules, which were selected out of a rulebase containing 20 rules. For a wizard, the script size was set to ten rules, which were selected out of a rulebase containing 50 rules. One or two default rules were added to the end of each script to ensure the execution of an action in case none of the rules from the rulebase could be activated.

4.5 The Experiments

The experiments aim at assessing the adaptive performance of an opponent party controlled by the dynamic scripting technique, against a player party controlled by static scripts. To quantify the relative performance of the opponent party against the player party, after each encounter for both parties we calculate the average of the fitness over the last ten encounters. If for the opponent party this number is higher, the opponent party *outperforms* the player party. We define the *average turning point* as the encounter after which the opponent party first outperforms the player party for at least ten consecutive encounters. Furthermore, we define the *absolute turning point* as the first encounter after which a consecutive sequence of encounters in which the opponent party wins is never followed by a longer consecutive sequence in which the opponent party loses. Low values for the turning points indicate good efficiency of dynamic scripting, since they entail that the opponent party, which uses dynamic scripting, needs only a few encounters to achieve the goal of outperforming an unchanging tactic, as used by the player party.

Four different static tactics were employed by the player party, namely the following: (1) strongly offensive, (2) disabling (freezing the enemies before attacking them), (3) cursing (hindering and weakening the enemies), and (4) strongly defensive. To assess the ability of the dynamic-scripting technique to cope with sudden changes in tactics, we defined three additional composite tactics: (1) random-party (which randomly picks one of the four basic tactics for each encounter), (2) random-character (whereby each character picks his own tactic independent from his comrades), and (3) consecutive-party (whereby a party keeps using one of the four basic tactics until it loses a fight, then switches to the next tactic).

For each of the basic tactics we ran 21 tests, and for each of the composite tactics we ran 11 tests. The results of these experiments are presented in the next section.

5 Results

Table 1 presents the results of the experiments described in section 4. We make the following three observations. The first observation is that the disabling tactic is easily defeated. Apparently the disabling tactic is not a good tactic, because it does not require adaptation of the rulebase to be dealt with. The second striking observation is that for

	Average Turning Point				Absolute Turning Point			
Tactic	Low	High	Avg.	Med.	Low	High	Avg.	Med.
Offensive	27	164	57	54	27	159	53	45
Disabling	11	14	11	11	1	10	3	1
Cursing	13	1784	150	31	4	1778	144	31
Defensive	11	93	31	23	1	87	27	18
Random Party	13	256	56	29	5	251	50	26
Random Char.	11	263	53	30	1	249	47	33
Consecutive	11	160	61	50	3	152	55	48

Table 1: Results of the experiments described in section 4. For each tactic the lowest, highest, average and median average and absolute turning points are shown.

both turning points the average is (in most cases) significantly higher than the median. The explanation is found in the rare occurrence of extremely high turning points. During early encounters chance can cause potentially successful rules to get a low rating or unsuccessful rules to get a high rating. As a result, the rulebase diverges from a good weight distribution to which it has trouble to return. Our technique has no mechanism to reduce the effect of early divergence, but it is clear that such a mechanism is needed to make dynamic scripting a useful technique. Third, the consecutive tactic, which is closest to human player behaviour, is overall the most difficult to defeat with dynamic scripting. Nevertheless, our dynamic-scripting technique is capable of defeating this tactic rather quickly.

6 Discussion

Our experimental results show that dynamic scripting is capable of adapting rapidly to static or changing tactics. Hence, dynamic scripting is efficient and fulfils the fourth requirement stated in section 3.

Although dynamic scripting turns out to be surprisingly efficient, the question remains whether it is sufficiently efficient for application in commercial games. For action CRPGs, such as DIABLO, the answer would be an unequivocal yes, because action CRPGs typically pit the player against hundreds of similar opponents. For more strategic CRPGs, such as BALDUR'S GATE, it depends on the definition of "similar opponents". While each opponent wizard in the game might be different, overall successful tactics for one wizard will also work for most other wizards. Furthermore, in our experiments we started our rulebase from scratch with identical weights for all rules. In a commercial release the rulebase would have been trained offline against pre-programmed scripts (cf. our experiments with the consecutive-party tactic). Confronted with standard tactics such a rulebase would adapt very quickly, while it still would have the ability to learn to generate good scripts to deal with novel tactics.

Finally, it should be noted that while our fitness criterion relied heavily on winning and losing encounters, in commercial games the fitness criterion should be different. In commercial games, the human player should (because of entertainment purposes) and will (because of saving and reloading functionalities) always win an encounter. In an actual commercial game fitness should rely more on the length of a fight and the amount of damage done. It might even be useful to punish a rulebase for winning a fight, so that the entertainment value of the game for weaker players is assured.

7 Conclusions

In this paper we proposed dynamic scripting as a technique to deal with online adaptation of opponent AI, suitable for complex commercial computer games such as CRPGs. Dynamic scripting is based on the automatic online generation of AI scripts for computer-game opponents by means of an adaptive rulebase. From our experimental results, we conclude that that dynamic scripting is fast, effective, robust, and efficient and therefore has the potential to be successfully incorporated in commercial games, although some more work must be done before the technique is ready to be implemented in actual commercial games.

References

[1] Mark Brockington and Mark Darrah. How *Not* to Implement a Basic Scripting Language. *AI Game Programming Wisdom* (ed. S. Rabin), pp. 548-554, Charles River Media, 2002.

[2] P. Demasi and A.J. de O. Cruz. Online Coevolution for Action Games. *GAME-ON 2002 3rd International Conference on Intelligent Games and Simulation* (eds. Q. Medhi, N. Gough and M. Cavazza), pp. 113-120, SCS Europe Bvba, 2002.

[3] John Manslow. Learning and Adaptation. *AI Game Programming Wisdom* (ed. S. Rabin), pp. 557-566. Charles River Media, 2002.

[4] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer Verlag, 2000.

[5] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Second Edition, Prentice Hall, Englewood Cliffs, New Jersey, 2002.

[6] Jonathan Schaeffer. A Gamut of Games. *AI Magazine*, Vol. 22, No. 3, pp. 29-46, 2001.

[7] Bob Scott. The Illusion of Intelligence. *AI Game Programming Wisdom* (ed. S. Rabin), pp. 16-20, Charles River Media, 2002.

[8] Pieter Spronck, Ida Sprinkhuizen-Kuyper and Eric Postma. Improving Opponent Intelligence through Machine Learning. *Proceedings of the Fourteenth Belgium-Netherlands Conference on Artificial Intelligence* (eds. Hendrik Blockeel and Marc Denecker), pp. 299-306, 2002.

[9] Paul Tozour. The Evolution of Game AI. *AI Game Programming Wisdom* (ed. S. Rabin), pp. 3-15, Charles River Media, 2002.

[10] Paul Tozour. The Perils of AI Scripting. *AI Game Programming Wisdom* (ed. S. Rabin), pp. 541-547, Charles River Media, 2002.

[11] Steven Woodcock. Game AI: The State of the Industry. *Game Developer Magazine*, August 2000.

A General View on Probabilistic Logic Programming

Joost Vennekens Sofie Verbaeten*

Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium

Abstract

Probabilistic logic programming offers a way of dealing with both relational and uncertain knowledge. As research in this field has lead to numerous formalisms, originating from diverse origins, it is not always easy to get a clear picture of this domain. In this work, we present a general "framework" for looking at such formalisms. We show how several actual formalisms fit into this view and how this leads to a clearer view on the relationships between them.

1 Introduction and Motivation

One way of dealing with both relational and uncertain knowledge, is by a combination of first order logic and probability theory. Research concerning this topic has not only lead to theoretical probabilistic logics, but also — analogous to logic programming and first order logic — to several more practical "probabilistic logic programming" formalisms. These have originated from a multitude of domains such as logic programming, machine learning and relational databases. However, relatively few contributions have attempted to clarify the relationships between these formalisms beyond the usual unmotivated "formalism X is more general than formalism Y"-claims.

In this work, we offer a first step towards remedying this situation. In section 2, we give an exact definition of our domain of discourse and offer a general overview. In section 3, we present a general description of a probabilistic logic programming "framework". In section 4, we then discuss three actual formalisms and show how they fit into the general framework and what information this can give us about the relationships between them. While research of this kind might not be highly innovative, we believe that, given the current state of the domain of probabilistic logic programming, it does constitute a useful contribution, perhaps even more so than the introduction of "yet another formalism".

*Sofie Verbaeten is a Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium)(F.W.O. - Vlaanderen).

2 Probabilistic Logic Programming

Before actually going into the details of the field of probabilistic logic programming, it is necessary to be precise about what, in this work, will be considered the definition of this domain. Here, we choose to define a probabilistic logic programming formalism (as opposed to a "probabilistic logic") as one in which each program specifies a *single* probability distribution over some logical entity. For instance, a typical type 2 approach (see below) will express a single probability distribution over a set of possible worlds, while a typical type 1 (see below) approach might express a single probability distribution over the Herbrand universe.

This definition can be motivated by considering the analogy with logic and logic programming. In general, the semantics of a logic (in the mathematical sense) is defined by a relation (usually denoted as $\models$) specifying which mathematical structures are considered models of a certain theory in this logic. Logic programming, however, could be seen as attempting to select a single "intended" model from all mathematical models (e.g. by considering only Herbrand models, making the closed world assumption, etc.).

Nevertheless, this definition is still rather arbitrary. For instance, it excludes, rather ironically, the first ever formalism to explicitly use this term: "Probabilistic Logic Programs" by Subrahmanian and Ng [6]. Indeed, this approach does not attempt to *define* a single probability distribution, but instead uses probability intervals to impose *constraints* on a set of possible distributions. Given the limited size of this contribution, it suits us to exclude such formalisms from consideration.

In the remainder of this section, we present a way to categorize probabilistic logic programming formalisms (as defined above). In [3], Halpern makes some important fundamental observations concerning first order logics of probability. In particular, he distinguishes two different types of semantics for such a logic: the so-called type 1 and type 2 semantics. Traditionally, the intuition behind this distinction is made clear by presenting the two following sentences.

- Type 1: "The probability that a randomly chosen bird flies, is greater than 0.9." (Randomness follows from the process of choosing a bird.)
- Type 2: "The probability that Tweety (a particular bird) flies, is greater than 0.9."(Randomness follows from our lack of knowledge.)

We can therefore make a distinction between formalisms representing the more "objective" type 1 knowledge and those representing the more "subjective" type 2 knowledge. In section 4.3, Stochastic Logic Programs [1, 5], a formalism which expresses type 1 knowledge, will be discussed.

Most of the work concerning probabilistic logic programming focuses on type 2 knowledge. To further categorize these remaining formalisms, we can look at whether they grew out of an attempt to extend logic programming with probability or one to construct a first-order version of a well-known propositional probabilistic framework. These last formalisms allow the representation of an entire "class" of propositional models, from which, for a specific query, an appropriate model can then be constructed "at run-time". Some examples of this approach, usually referred to as Knowledge Based Model Construction, are: Bayesian Logic Programs

of Kersting and De Raedt [4] which will be discussed in section 4.2 and Probabilistic Relational Models by Getoor et al [2].

Finally, the type 2 extensions of logic programming contain formalisms such as Poole's Independent Choice Logic [7], which we will discuss in section 4.1, and Programming In Statistical Modeling by Sato et al [8].

3 General framework

Having defined our domain of discourse, we now give a general description of such a formalism, into which we will, in section 4, fit several actual approaches. Since, as any relevant textbook will show, the concept of the *experiment*[1] lies at the heart of probability theory, it will also feature prominently in our discussion. Indeed, taking such an experiment-centric view, we can formulate the aim of probabilistic logic programming (as defined in section 2) as that of allowing the definition of a single complex experiment (the semantics of the entire program) (e.g. that of determining which of a set of possible world is the "real" one) in terms of a number of smaller experiments (the semantics of some "part" of the program).

More specifically, the semantics of some probabilistic logic program defines a set of experiments. In general, this set could be defined inductively, i.e. the set of possible outcomes or their probability is allowed to depend on the results of some "previous" experiments. Each result of such an experiment has, by definition, a probability. The semantics of the formalism then offers a "combination function"[2], which specifies the probability of each "total result" (i.e. a result for each experiment), in terms of the probabilities of the results of individual experiments. This yields a combined probability space, which then, finally, is mapped into some sort of result space, which fully defines the semantics of the formalism.

As such, the semantics of a probabilistic logic programming formalism can be completely described by stating which experiments it defines, how the probabilities of these simple experiments are combined and how the resulting probability space is translated to a result space.

4 A Detailed Comparison

In this section, we discuss a single formalism chosen from each of the categories presented in section 2. We have based this choice both on our own opinion of which formalism best represents an entire approach and on the availability of literature comparing the chosen formalisms to others; [4] compares BLPs to a number of other formalisms, while [1] does the same for SLPs. We will give a description of each of these formalisms, show how they fit in the terminology of section 3 and attempt to clarify the relationships between them. To illustrate this, we will also represent the Hidden Markov Model (HMM) of figure 1 in each of these formalisms.

[1] By "experiment", we simply mean a probability distribution over some set of possible results.

[2] To avoid confusion, it should be noted that our use of this term is unrelated to combination functions in the context of Bayesian networks.

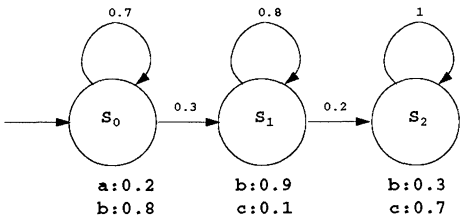


Figure 1: A Hidden Markov Model

4.1 Independent Choice Logic

The Independent Choice Logic (ICL) [7] originated as a probabilistic generalization of abductive logic programming. A program in this logic consists of two parts: A normal logic program F and a set C of declarations of following form $random([a_1 : \alpha_1, \ldots, a_n : \alpha_n])$, with a_i atoms (sometimes called hypotheses or abducibles) and $\alpha_i \in [0,1]$, such that $\sum \alpha_i = 1$. Furthermore, such a pair (C,F) must satisfy following conditions: F must be acyclic, no atom a_i appearing in a *random*-statement in C may unify with another atom a_j which also appears in a *random*-statement in C and no head of a clause in F may unify with an atom a_i appearing in C.

The semantics of an ICL program is defined using the concept of a "total choice", which is simply a selection of an atom a_{ij} from each ground instantiation of a *random*-declaration j. Each such choice corresponds to a possible world, namely the unique stable model of F augmented with the chosen facts a_{ij}. Since total choices are mutually exclusive, there is only one choice for each possible world. The probability of a world is $\prod_j \alpha_{ij}$, where α_{ij} is the probability associated with atom a_{ij} chosen from *random*-declaration j.

So, rephrasing this in our terminology, each *random*-statement defines a single experiment, namely that which chooses an atom from its argument-list. The probabilities α_{ij} are combined into a probability for each total result by assuming independence between the simple experiments, i.e. the "combination function" used is simple multiplication. This combined probability space is than transformed into the "result space", by mapping each set of choices to the unique stable model of the clauses F and the chosen facts.

This discussion shows that, in modeling the HMM of figure 1, the state transitions and possible output will need to be represented by *random*-statements, while the structure of the HMM will be modeled in the logical part:

$$state(S,T) \leftarrow state(Sprev, T-1),\ goto(Sprev, S, T-1).$$
$$out(C,T) \leftarrow state(S,T),\ out(S,C,T).$$
$$state(s0,0). \qquad goto(s2,s2,T).$$
$$random([goto(s0,s1,T) : 0.3, goto(s0,s0,T) : 0.7]).$$
$$random([goto(s1,s2,T) : 0.2, goto(s1,s1,T) : 0.8]).$$
$$random([out(s0,a,T) : 0.2, out(s0,b,T) : 0.8]).$$
$$random([out(s0,b,T) : 0.9, out(s0,c,T) : 0.1]).$$

$random([out(s0, b, T) : 0.3, out(s0, c, T) : 0.7])$.

The interesting part of this formalism, is that the probabilistic and logical part serve completely separate purposes: The *random*-statements define a very simple chance-setup and the logical part serves only to derive the final possible worlds.

4.2 Bayesian Logic Programs

Bayesian Logic Programs [4], or BLPs for short, are based on the (propositional) concept of Bayesian networks. Each ground atom a with predicate-symbol p represents a random variable, which can take on a value from a domain d_p associated with this predicate p. A clause in a BLP is of the form $H \mid A_1, \ldots, A_n$, with H and the A_i atoms. Such a clause requires a conditional probability table (CPT) to be associated with it, specifying the conditional probability of each possible value for the random variables corresponding to the atom in the head, given the values of those corresponding to the atoms in the body. For each predicate p, there is at most one clause with p in its head[3].

The semantics of a BLP is defined by relating it back to a Bayesian network. As previously mentioned, each element of the Herbrand base corresponds to a node. There is an edge from a node representing ground atom X to a node representing ground atom Y if the program contains a clause $c = X' : - \ldots, Y', \ldots$, for which a substitution θ exists, such that $X'\theta = X$ and $Y'\theta = Y$. The CPT associated with such an edge is simply that which was associated with clause c. In order for this semantics to work, this Bayesian network must obviously be well defined, i.e. the number of parents of each node must be finite and, just as in ICL, the program must be acyclic.

So, each clause in a BLP defines a simple experiment, namely that of choosing one possible values for the predicate of the atom in its head. The probability distribution on these possible outcomes obviously depends on the outcome of "previous" experiments, i.e. those corresponding to the atoms in its body. This explains the necessity of the acyclicity condition. The probabilities of these simple experiments are then combined by making the usual independence assumptions of Bayesian networks. The resulting combined probability space gives the semantics of the BLP.

Modeling the HMM of figure 1 by a BLP, is rather straightforward:

$out(T) \mid state(T)$.
$state(T) \mid state(T-1)$.
$state(0)$.

$out(T)$	$state(T) = s_0$	$\cdot = s_1$	$\cdot = s_2$
a	0.2	0	0
b	0.8	0.9	0.3
c	0	0.1	0.7

[3]Technically speaking, a BLP can have multiple clauses for a certain predicate. However, in this case, a combination rule must be given, which specifies how these clauses can be combined to a single clause.

state(0)	
s_0	1
s_1	0
s_2	0

$state(T)$	$state(T-1) = s_0$	$\cdot = s_1$	$\cdot = s_2$
s_0	0.7	0	0
s_1	0.3	0.8	0
s_2	0	0.2	1

Comparing this discussion to that of ICL in section 4.1, we see that the mayor difference between these two formalism lies in the fact that the logical part of a BLP structures the set of experiments it describes (and therefore BLPs use conditional probabilities), while ICL describes a "flat" set of experiments and uses its logical part only in mapping the combined probability space to a result space.

In transforming a BLP to an ICL program, we need one *random*-statement for each possible experiment in the BLP, i.e. one for each possible assignment of values to the atoms in the body of each clause. Therefore, we need to make the value of an atom, which is basically an implicit argument in a BLP, explicit. We then split a BLP clause into a set of clauses, namely one for each possible instantiation of the extra arguments. A such, each of these new clauses needs to describe exactly one *unconditional* ICL experiment. We can accomplish this, by splitting such a clause further, i.e. creating a new clause for each possible instantiation of the extra argument of the atom in its head and differentiating the bodies accordingly by adding a new artificial atom to them. It then suffices to provide an appropriate *random*-statement for these new atoms. In this way, these artificial atoms allow us to simulate the conditional probabilities of a BLP by a set of independent experiments.

Conversely, a definite ICL programs can also be transformed to a BLP. The translation of a *random*-statement is straightforward: For a *random*-statement with n choices, we create a new atom with domain $1, \ldots, n$ and add clauses with degraded CPTs, stating that if this new atom takes on value i then (with probability 1) the ith element from the *random*-statement holds. The logical part of an ICL program can be modeled by giving each of the atoms a domain $\{true, false\}$ and using degraded CPTs to simulate the logical semantics of the clauses, i.e. the head is true if and only if all atoms in the body are true.

4.3 Stochastic Logic Programs

Stochastic Logic Programs [1], or SLPs for short, are a probabilistic generalization of stochastic context free grammars, which "generate" SLD refutations. A SLP consists of a set of range-restricted definite Horn clauses. Each clause c has a label $\lambda_c \in [0, 1]$, which specifies the probability that c is used in an SLD-refutation when an atom with the same predicate as that in the head of c needs to be resolved. As such, denoting by C_p the set of all clauses with a predicate symbol p in the head, the condition that $\sum_{c \in C_p} \lambda_c = 1$ is imposed[4].

SLPs represent type 1 knowledge. More precisely, the probability of each ground instantiation $p(a_1, \ldots, a_n)$ of a predicate p/n is defined as the sum of the probabilities of each proof for $p(a_1, \ldots, a_n)$, normalized by the sum of the

[4]For simplicity, here we only consider "normalized, pure SLPs".

probabilities of each proof for $p(X_1, \ldots, X_n)$. In this formula, the probability of a proof using clause c a number of $t(c)$ times is $\prod_c \lambda_c^{t(c)}$.

In a SLP, a simple experiments is the selection of one clause from all clauses containing a certain predicate in the head. There is one such experiment for each choice-point encountered in a SLD-derivation of a certain query. Therefore, such an experiment obviously depends on the results of previous experiments. The probabilities of simple experiments are once again combined by simple multiplication. The step of going from the combined probability space on SLD-derivations to a result space of possible instantiations of the variables in the query, is done by summing the probability of all refutations which lead to a particular instantiation and then normalizing these sums.

The following SLP gives, for the HMM of figure 1, the desired distribution on the instantiation of the *output*/2 predicate. Note that here we do not need to use a third argument (the time-point) for *out*/2 and *goto*/2.

$1 : output(C,T) : -state(S,T), out(C,S).$
$0.5 : state(S,T) : -T > 0, state(Sprev,T), goto(Sprev,S).$
$0.5 : state(s0,0).$

$0.2/3 : out(a,s0).$	$0.9/3 : out(b,s0).$	$0.3/3 : out(b,s0).$
$0.8/3 : out(b,s0).$	$0.1/3 : out(c,s0).$	$0.7/3 : out(c,s0).$
$0.7/3 : goto(s0,s0).$	$0.8/3 : goto(s1,s1).$	$1/3 : goto(s2,s2).$
$0.3/3 : goto(s0,s1).$	$0.2/3 : goto(s1,s2).$	

Transforming a SLP to an ICL-program, requires constructing a *random*-statement for each possible SLD-choice-point. Therefore, we need to add an extra argument to the atoms, allowing us to differentiate between different "calls" of the same atom. Then, we need to add a *random*-statement for each predicate, containing an atom for each clause with that predicate in its head. These atoms are added to the bodies of the corresponding clause, such that this clause can only by "used" if it is chosen. Finally, the sum of the probabilities of all worlds which contain a specific ground instantiation of a predicate p/n, divided by the sum of the probabilities of all worlds containing *some* instantiation of p/n, gives the semantics of the SLP.

Due to SLPs type 1 nature and the tight coupling of its semantics to SLD resolution, it is currently not clear whether and how one can transform ICL-programs to SLPs.

5 Conclusions and Future Work

In section 3, we presented a general "framework" for probabilistic logic programming formalisms. In section 4, we showed how a probabilistic generalization of abductive logic programming (ICL) and a first-order generalization of Bayesian networks (BLP) and of stochastic context-free grammars (SLP) fit into this view. This allowed us to see past superficial differences in terminology and presentation and clearly determine the differences and similarities between them. ICL uses a

simple set of experiments, but has a complex mapping from combined probability space to its "result space". A BLP defines a more complex set of experiments, but its semantics is simply the combined probability space. A SLP also has a complex set of experiments and, in addition to this, its semantics maps the combined probability space to a distribution over instantiations of the query, giving it its typical type 1 characteristics. In combining the probabilities of simple experiments, all these formalisms resort to assuming independence by default. We were also able to study equivalences between these formalism. ICL turned out to be the most general, with it being possible to represent both BLPs and SLPs in this formalism. Conversely, definite ICL programs can be transformed to BLPs, but the situation with SLPs is less clear.

While we could not, due to space restrictions, make this framework or our comparison of the formalisms mathematically precise, nor offer a fully complete view of the domain, we do believe that the work presented here already shows that doing so in future work would offer an even more significant contribution.

References

[1] J. Cussens. Integrating probabilistic and logical reasoning. *Electronic transactions on Artificial Intelligence*, 3(B):79–103, 1999.

[2] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*, pages 7–34. Springer-Verlag, 2001. to appear.

[3] J.Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1989.

[4] K. Kersting and L. De Raedt. Bayesian logic programs. In J. Cussens and A. Frisch, editors, *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pages 138–155, 2000.

[5] S. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Proceedings of the 5th International Workshop on Inductive Logic Programming*, page 29. Department of Computer Science, Katholieke Universiteit Leuven, 1995.

[6] R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[7] D. Poole. The Independent Choice Logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):7–56, 1997.

[8] T. Sato and Y. Kameya. PRISM: A language for symbolic-statistical modeling. *Proceedings of IJCAI 97*, pages 1330–1335, 1997.

On Heuristics for Learning Model Trees

Celine Vens Hendrik Blockeel

Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract

Induction of decision trees is a popular learning technique, not just for classification but also for numerical prediction (regression). The term "model trees" is commonly used for trees that in their leaves contain some (usually linear) regression model. Popular implementations of model tree learners use reduction of variance as a heuristic for selecting tests during the tree construction process. In this paper, we show that systems employing this heuristic may exhibit pathological behaviour in some quite simple cases. This is not visible in the predictive accuracy of the tree, but it reduces its explanatory power. We propose an alternative heuristic that yields equally accurate but simpler trees with better explanatory power, and this at little or no additional computational cost.

1 Introduction

Induction of decision trees [7, 1] is a frequently used machine learning technique, not just for classification but also for numerical prediction (regression). Among regression trees, we can distinguish those that with each leaf associate a constant, as induced by for instance CART [1], and those that associate a less trivial, usually linear, model with each leaf. For the latter we will use the term "model trees". A popular algorithm for building model trees is M5 [8], of which M5' [9] is a reimplementation that is included in the well-known Weka software [10]. Another well-known approach is RETIS [3], which uses a more sophisticated heuristic.

The main difference between the M5' and RETIS approaches is that M5' first learns a standard regression tree (with constants in the leaves) and only afterwards, during a pruning phase, turns it into a model tree. RETIS aims immediately at building a model tree and uses a heuristic tuned towards this task. This heuristic, however, is quite expensive to compute, which may render the RETIS approach infeasible for certain practical problems.

M5', on the other hand, has its deficiencies too. In Section 2 we discuss the M5' approach to model tree building and demonstrate its potential to exhibit pathological behaviour that strongly reduces the explanatory power of the induced trees. This motivates us to look for a heuristic that is approximately as efficient as that of M5', but avoids this behaviour. We propose such a heuristic in Section 3. Section 4 presents empirical results, comparing our approach to existing ones. The results are positive for synthetic data, but somewhat ambiguous for real data sets. We conclude in Section 5.

2 The M5' Approach

Existing regression tree algorithms are instantiations of the "top-down induction of decision trees" algorithm (TDIDT); see Quinlan [7] or Breiman et al. [1] for full details on the TDIDT method. Roughly, the method recursively partitions a set of data such that "maximally homogeneous" (with respect to some target attribute) subsets are obtained. These subsets are associated with specific attribute values.

TDIDT algorithms vary mainly with respect to a.o. the heuristic they use to decide which partitioning is best and the model that is stored in each leaf. For instance, CART [1], when building regression trees, uses variance reduction as a heuristic (that is, it tries to reduce the variance of the target variable within the subsets as much as possible), and the model stored in a leaf is the mean of the target values of all examples in that leaf. M5' [9] also uses variance[1] reduction as a heuristic, and fills in constants in the leaves, but in its pruning phase it changes internal nodes into leaves containing a linear model if that model performs at least equally good as the subtree rooted in that node. The linear model uses as predictor variables only attributes occurring in the subtree that it replaces.

Keeping in mind that linear models will be built in the subsets, it seems reasonable to select tests that maximize the expected quality of these models. In that case, variance reduction is not a very suitable heuristic. As also noted elsewhere [3, 4], the quality of a linear model constructed for a data set is quite independent of the variance of that data set. Given this independence, there is no reason why variance would perform better than random splitting. In fact, it may even be worse, as the following simple example shows.

Consider the piecewise linear function $y = x, 0 \leq x \leq 1$; $y = 2 - x, 1 < x \leq 2$ (see the left part of Figure 1). Clearly the simplest model tree to represent this function is one that splits on the condition $x < 1$ and builds two linear models.

In the following we consider a continuous uniform distribution of (x, y) couples. Note that for the optimal tree, the variance along y of both subsets is equal to the variance along y of the original set; thus, from the point of view of reducing variance, this tree would not be considered a good tree. The split $x < c$ that is found by M5' is the one that minimizes the sum of the weighted variance over both subsets. For a continuous uniform distribution, the optimal c can be computed by minimizing $h(c) = c \cdot Var(y|x \leq c) + (2 - c) \cdot Var(y|x > c)$, where the factors c and $2 - c$ are weights referring to the respective sizes of the subsets and the Var factors are the variances of these subsets. The optimal c is then defined as $c^* = \arg\min_c h(c)$. Due to symmetry, $2 - c^*$ is an optimum in the interval [1,2].

The function $h(c)$ is plotted in Figure 1, and there it can clearly be seen that a minimum is obtained near 0.4; that is, quite far away from the optimal split point 1. In fact, even random splitting can be expected to yield slightly better results on average, since it has 60% probability of creating a split closer to 1.

Without loss of generality, given the symmetry, we assume $c^* < 1$. The left subset yields a perfect linear model. However, as the model is not actually constructed as part of the stopping criterion, M5' constructs a subtree for this subset,

[1] Or variations of variance, such as standard deviation [9] or the 5th root of the variance, as in the Weka implementation.

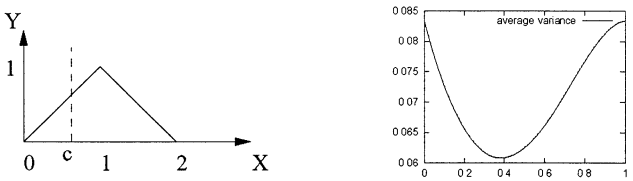


Figure 1: Left: a piecewise linear function and split point c. Right: the weighted mean of the variance in the subsets defined by $x \leq c$ and $x > c$ as a function of c.

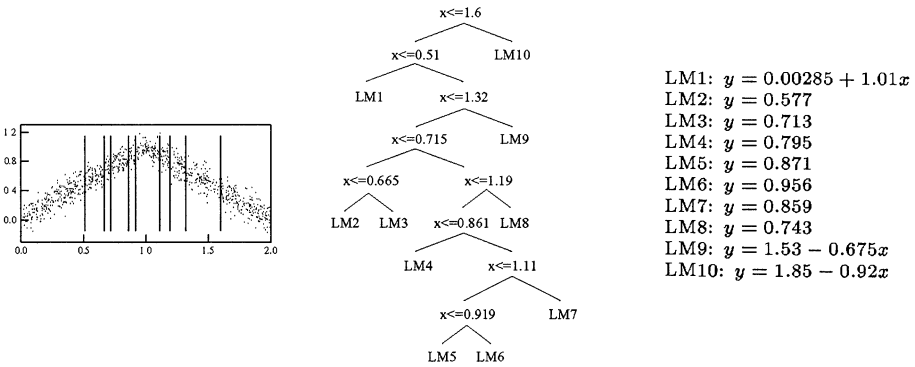


Figure 2: A simple data set, and the model induced for it by a variance based model tree builder. The vertical lines in the data plot indicate the split points.

partitioning this linear area into many small areas with ever decreasing variance; only in the pruning phase does it remove this subtree to build the linear model. The right subset is a function similar to the original one but somewhat more asymmetric. Our experiments confirm that similar behaviour can be expected (see further). Note that in contrast to the left branch, the right branch cannot be pruned to a single leaf; and that also further down the tree, the splits will always be off the optimal point. Thus a relatively deep tree will still remain.

The above reasoning shows that variance-based heuristics tend to split the data set in the wrong places. This reduces the explanatory power of the tree: first, a larger tree is obtained than is necessary, and second, the splits in the tree are not really informative. Moreover, superfluous partitioning of the data space causes relatively small areas to be separated, with each area containing few examples. As local models built from few examples tend to be less accurate, the predictive performance of the induced trees may also be reduced because of this.

In our experimental section we further explore this behaviour of M5'. At this point, we just illustrate the concrete behaviour of M5' on the simple function mentioned above, for a random sample of data points with some noise added. Figure 2 shows the tree M5' builds for such a data set. It confirms our theoretical analysis: the model built by M5' is good in the sense that it has a reasonably good predictive performance, but its explanatory power is diminished.

3 A Simple Linear Regression Based Heuristic

Our simple case not only shows that variance does not work very well as a heuristic, it also suggests that constructing linear models for the subsets and evaluating their goodness of fit should work fine. This is exactly the approach followed by RETIS [3], another TDIDT instantiation. RETIS evaluates a split by building a multiple linear model for each subset and computing its residual variance. Also in the leaves multiple linear regression models are built. Unfortunately, these multiple regressions result in a complexity that is cubic in the number of attributes [4], which is too high for many practical applications.

Another example of a multiple linear regression system is SMOTI [4]. It induces an alternative kind of decision tree, where besides splitting nodes, regression nodes can be introduced. The complexity is quadratic in the number of attributes.

The simple case described above deals with a single predictor variable, which is uncommon. Consequently, the variable used to define the split is the same as the variable used in the regression. When multiple predictor variables exist, a number of different approaches are possible. We characterize each of them by listing the variables for which it performs regression *in order to evaluate the split* (not to be confused with the construction of linear models in the leaves), given the variable it splits on. We consider only univariate splits here. All these options could be extended by considering also multivariate splits [2], but this extension is more or less orthogonal to the dimension discussed here, and hence out of scope for us.

The four options we distinguish are:

1. no regression (the M5' approach)
2. simple regression on the split attribute if it is numerical
3. simple regression on all numerical attributes separately
4. multiple regression on all numerical attributes together (RETIS approach)

Our reason for listing exactly these different options is that they differ significantly with respect to their computational complexity: Options 1 and 2 are linear in the number of attributes, Option 3 quadratic and Option 4 cubic. All of these complexity factors are multiplied with at least N, with N the number of examples.

Our motivation for considering Option 2 is threefold:

- It has a computational complexity close to that of Option 1, differing only with a constant factor[2]. In fact, since we hope to build smaller trees, the slightly more complex computation might well be compensated.

[2]Variance, $\sigma^2 = \sum_i (y_i - \bar{y})^2/n$ with $\bar{y}$ the mean of the y_i, can be computed entirely from the sufficient statistics $\sum_i (1, y_i, y_i^2)$ with i varying over all elements of the data set. Similarly, a simple linear regression model as well as its residual variance can be computed from $\sum_i (1, x_i, x_i^2, y_i, y_i^2, x_i y_i)$. This takes about three times longer. Computing the residual variance from these statistics is about four times as much work as computing the total variance from them. Hence, our heuristic is roughly three to four times as expensive to compute as the original one.

- It is consistent with an idea that underlies most univariate decision tree methods: evaluating the predictive power of each attribute independently. Option 3 boils down to splitting on one attribute and examining how this influences the predictive power of other attributes; this is similar to the use of lookahead in decision tree building, an approach advised against [6].
- It provides a solution for the undesirable behaviour that systems using Option 1 exhibit. More specifically, it should work better in those cases where at least one variable has an influence on the target variable that is non-linear but can be approximated with a piecewise linear function.

Obviously, for each of the Options 1-3 it is possible to point out situations where the next more complex option performs better. The point that we wish to make, is that the move from Option 1 to 2 solves at least some problems, and costs almost nothing with respect to efficiency.

The model-tree induction algorithm we propose follows Option 2 and is a variant of M5' that we call MAUVE ("M5' Adapted to use Uni-VariatE regression"); it only differs with respect to the heuristic, which is based on simple regression.

4 Experimental Results

The MAUVE implementation used is identical to the M5' implementation in Weka, except for the heuristic, as mentioned above. Standard settings in Weka were used, except for smoothing[3], which was switched off.

Our experimental section first discusses experiments on synthetic data sets, and then experiments on real world data sets. The relevant criteria are predictive performance, tree size, and induction time. Predictive accuracies are estimated using the mean RRMSE[4] of ten tenfold cross-validations.

4.1 Synthetic Data Sets

Figure 3 shows some simple functions. For each function $f(x)$, a data set was constructed by drawing a random sample of 1000 values from a uniform distribution over x, and associating with each x a $y = f(x) + \epsilon$ with ϵ a normally distributed random variable with mean 0 and standard deviation σ that represents noise. Varying σ had no significant influence on the interpretation of our results.

For each function, we report the partitioning created by model tree builders using simple regression, variance, and random splits. These results confirm our earlier findings: for a variety of functions, MAUVE tends to find simpler models, with fewer split points that intuitively make more sense. Introducing random splits works approximately as well as using variance-based splits, or even slightly better.

[3]Smoothing [8] is a method for improving predictions of a model tree by taking a weighted average between models in the leaves and linear models higher up in the tree. It changes the interpretation of a model tree somewhat, which would somewhat complicate the interpretation of our results. However, with smoothing the results are not significantly different.

[4]RRMSE = root relative mean squared error, this is the root of the ratio of the mean squared error of the tree to the mean squared error of a trivial model always predicting the mean.

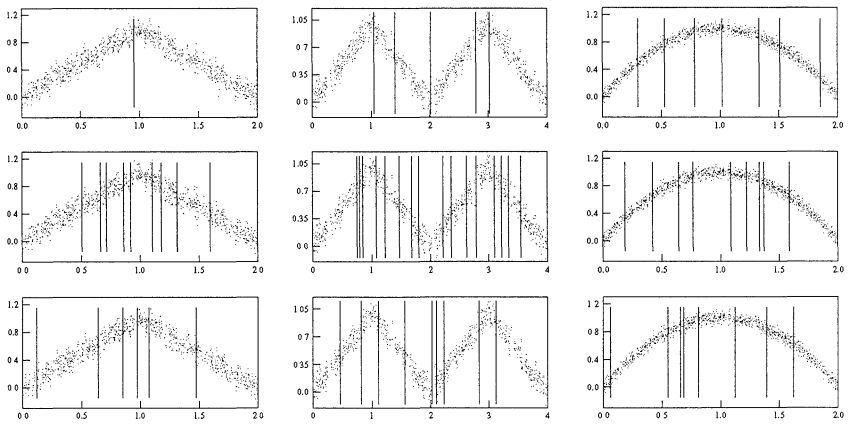


Figure 3: Approximating functions of one variable. Three different functions are shown: (from left to right) function 1, function 2 and function 3; for each we indicate the thresholds that are created by (from top to bottom) MAUVE; M5' with its original variance heuristic; M5' with random splits.

Table 1: Predictive accuracy, model tree size and induction times when using simple regression (MAUVE), variance (M5'), and random heuristics (Rnd).

	Accuracy (RRMSE)			Size (leaves)			Time (seconds)		
	MAUVE	M5'	Rnd	MAUVE	M5'	Rnd	MAUVE	M5'	Rnd
function 1	33.99%	34.60%	34.21%	2	10	6	0.11	0.13	0.11
function 2	33.05%	34.33%	33.36%	6	17	10	0.09	0.12	0.09
function 3	17.05%	17.14%	17.13%	8	10	9	0.09	0.12	0.09
function 4	20.47%	23.35%	56.47%	4	14	17	0.08	0.13	0.06

We have also constructed piecewise linear functions in multiple predictor variables. Figure 4 presents a representative example problem. Again we see that MAUVE performs better than M5'.

Some statistics on the induced trees are presented in Table 1. They confirm that the models induced by MAUVE are always simpler without causing any loss of accuracy. Moreover, the induction times for MAUVE are similar to those of M5', which suggests that the tree building procedure stops earlier and that this effect compensates for the more complex heuristic.

4.2 Real World Data Sets

We have compared the performance of the different heuristics on a small number of UCI data sets [5] that have been used in the literature on model trees [8, 9, 4]. The results are much less convincing here. Table 2 summarizes the results.

On average Mauve does not perform better, nor worse, than M5', with respect to accuracy and size (the differences are not statistically significant). Time-wise there is an improvement in all cases, even where Mauve's tree size is larger.

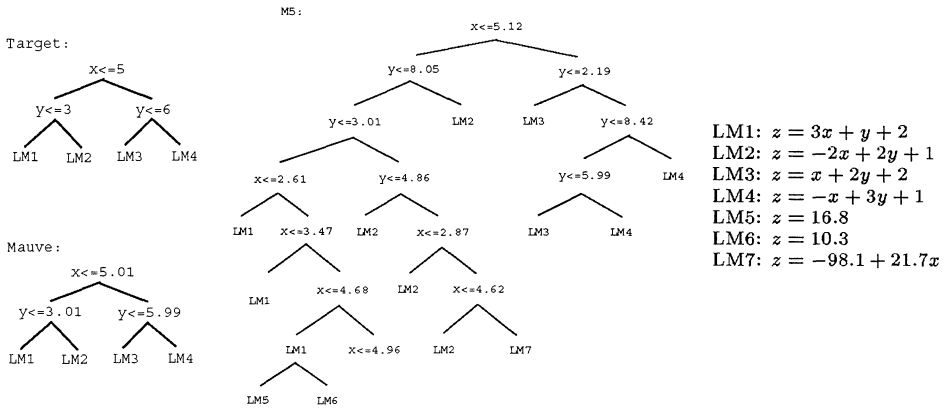


Figure 4: Learning a piecewise linear function of two variables (function 4). The original function is shown, and how it is approximated by MAUVE and by M5'.

Table 2: Results for UCI data sets.

	Accuracy (RRMSE)		Size (leaves)		Time (seconds)	
Data set	Mauve	M5'	Mauve	M5'	Mauve	M5'
MachineCPU	40.21%	42.48%	5	3	0.03	0.04
Abalone	67.71%	66.43%	29	12	1.62	2.55
Auto-Mpg	41.50%	36.49%	3	4	0.14	0.30
Auto-Price	40.30%	40.84%	2	8	0.05	0.10
Housing	58.21%	44.43%	18	18	0.16	0.27
Diabetes	93.48%	93.17%	1	3	0.002	0.003

These results suggest that the real datasets used in this experiment do not exhibit the behaviour that was introduced in the artificial datasets (namely, that some variables have a piecewise linear influence on the target variable). Further study will be needed to investigate how representative, in general, these artificial datasets are. The experiments also suggest, however, that adopting the new heuristic entails little risk: there are no cases where it performs significantly worse.

5 Conclusions

The contributions of this paper can be summarized as follows. First, we have studied the behaviour of variance as a heuristic for building model trees. While the inappropriateness of variance for this task, in itself, was pointed out in earlier work [3, 4], no investigation was performed on exactly how this influences the quality of the induced trees. Our results show it mainly influences the explanatory power of the tree, rather than its predictive power. Since explanatory power is often mentioned as an important advantage of trees over black-box models, it is worthwhile to try to improve it.

Second, we have presented an approach that does indeed induce model trees with better explanatory power. This approach is linear in the number of attributes, and as such differs from the complexity of the most efficient approaches only with a

constant factor. Currently existing alternative approaches towards building better trees (such as RETIS or SMOTI) have a complexity that is at least quadratic.

Our experimental validation on synthetic datasets confirms that the alternative heuristic we propose tends to yield simpler trees with equal predictive accuracy, without a significant loss of efficiency. Experiments on a small number of real-life datasets, however, are more ambigious, which suggests that the properties that are introduced in our synthetic datasets (namely, that variables have an influence on the target variable that can be approximated well with a piecewise linear function) may not hold for these data sets. In this case, neither a clear improvement nor a clear deterioration of performance, as compared to M5', was found.

References

[1] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, 1984.

[2] C.E. Brodley and P.E. Utgoff. Multivariate decision trees. *Machine Learning,* 19:45–77, 1995.

[3] A. Karalic. Employing linear regression in regression tree leaves. In *European Conference on Artificial Intelligence,* pages 440–441, 1992.

[4] D. Malerba, A. Appice, M. Ceci, and M. Monopoli. Trading-off local versus global effects of regression nodes in model trees. In H.-S. Hacid, Z.W. Ras, and Y. Zighed, D.A. Kodratoff, editors, *Foundations of Intelligent Systems, 13th International Symposium, ISMIS'2002,* volume 2366 of *Lecture Notes in Artificial Intelligence,* pages 393–402. Springer-Verlag, 2002.

[5] C.J. Merz and P.M. Murphy. UCI repository of machine learning databases [`http://www.ics.uci.edu/~mlearn/mlrepository.html`], 1996. Irvine, CA: University of California, Department of Information and Computer Science.

[6] S.K. Murthy and S. Salzberg. Lookahead and pathology in decision tree induction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence,* pages 1025–1031. Morgan Kaufmann, 1995.

[7] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann series in machine learning. Morgan Kaufmann, 1993.

[8] J.R. Quinlan. Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence,* pages 343–348. World Scientific, Singapore, 1992.

[9] Y. Wang and I.H. Witten. Inducing model trees for continuous classes. In *Proc. of the 9th European Conf. on Machine Learning Poster Papers,* pages 128–137, 1997.

[10] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.

Plan Merging: Experimental results

Mathijs de Weerdt, Roman van der Krogt and Jonne Zutt

Delft University of Technology,
PO Box 5031, 2600 GA Delft, The Netherlands
{M.M.deWeerdt | R.P.J.vanderKrogt | J.Zutt}@ewi.tudelft.nl

Abstract

In this paper we discuss the results of a *plan merging* algorithm. This algorithm coordinates the plans of multiple, autonomous agents, each able to independently find a plan. This algorithm is evaluated using realistic data from a taxi company. We show that when we allow passengers to be a few minutes later at their destination and share rides, we can obtain more than 5% reduction of the taxi driving distance. When we allow for a delay of 15 minutes (a common amount of time in subsidized transport) we can gain up to 30%.

1 Introduction

Many planning and coordination problems can be modeled as a multi-agent planning problem, i.e., a (distributed) set of related AI-planning problems. For example, the coordination of transport organizations, armies, production processes, etc., to name just a few. One of the techniques to solve such multi-agent planning problems is *plan merging*. Using this approach, we assume that each planner is capable of finding a solution for its own planning problem, for example using a refinement planning method [5]. When all agents have constructed their plans, these plans are compared to see whether cooperation is beneficial. For example, if during a plan two taxis bring someone from the train station to the hospital at about the same time, these passengers can share a taxi.

This paper discusses the results of a plan merging approach using so-called *resource facts* (part of our Action Resource Formalism [9, 10]). Such a resource fact aggregates the attributes of exchangeable objects into a single predicate, facilitating the search for opportunities to cooperate.

In the next section we take a closer look at the plan merging problem and its solution. Then, we present experimental results obtained from using this plan merging algorithm on data on the operations of taxi company. We finish by discussing possible extensions to the plan merging algorithm and related work.

2 Plan merging

The principle idea behind plan merging is that agents can refrain from work (i.e. they can remove actions from their plans) for which the results (i.e. the resources deliverd by these actions) are readily available at (several) other agents. Given

some plan for an agent that satisfies its goals, this agent can often reduce more costs by cooperating with other agents than by trying to optimize locally. In case multiple agents have similar parts in their plans, this can be quite advantageous. In this section we describe how this can be done by exchanging resource facts using *plan merging* [10].

To facilitate the exchange of resource facts, we assume one of the agents, or a trusted third party, acts as the *auctioneer*. The auctioneer announces when the agents can start the plan merging, thereby announcing some minimum allowed cost reduction value. All agents deposit requests with this auctioneer. Each *request* corresponds with the removal of an action from an agent's plan and contains a set of resource facts the agent needs to remove the particular action. Further, the request contains a cost reduction value defined by the difference in costs between the old plan and the resulting plan if the exchange succeeds.

The (greedy) auctioneer deals with the request with the highest potential cost reduction first. We assume all the agents honestly announce their cost reduction values. Right before each auction round starts, the requesting agent (a_i) is asked for the specific set of resource facts that has to be replaced by resource facts of other agents – this set is called the *RequestSet*. This set is not necessarily equal to the set in the initial request, since other exchanges influence the availability of resource facts for the agents.

To put up an auction for a request of an agent a_i, the set of requested resource facts is sent to each agent, except to a_i. The agents return all their free resource facts for which there is an equivalent one in the request set *RequestSet*, and include the price of each of their offered resource facts. When all bids (collected in R') are collected by the auctioneer, it selects for each requested resource fact the cheapest bid.

If for each resource fact in *RequestSet* a replacement can be found, the auctioneer tells the requesting agent a_i that it may discard the corresponding action(s). The replacing resource facts R'' are marked as goals for the providing agents, and become additional 'initial' resource facts for agent a_i. Furthermore, we have to add *dependencies* between these goals of the providing agents and the initial resource facts for the requesting agent.

If an agent's request succeeds, it discards the corresponding action in its plan, together with a possible set of actions in front of this particular action that also becomes obsolete. At the end of each successful exchange each involved agent has to update the cost reduction values of all of their requests, because this value can change as the agent can now have more or less resource facts available. One could repeat this process until none of the auctions has been successful.

The plan-merging algorithm is an *any-time* algorithm, because it can be stopped at any moment. If the algorithm is stopped, it still returns an improved set of agent plans, because this algorithm used a greedy policy, i.e., dealing with the requests with the largest potential cost reduction first. Algorithm 1 can be shown to have a worst-case time complexity of $O(n^2)$ where n is the number of actions of the plans of all agents involved in plan merging [10]. In this paper, we focus on the experimental results of the plan merging algorithm and give only a short introduction to the formalism and the plan merging algorithm itself. For a more elaborate

discussion of both the formal and practical analysis of this algorithm we refer to the thesis by De Weerdt [9].

Algorithm 1 PLAN_MERGING(A)

1. *auctioneer broadcasts minimum allowed cost reduction.*
2. *auctioneer retrieves requests with their cost reduction from all agents A.*
3. **while** *some requests left* **do**
   - 3.1. *get the request with the highest cost reduction.*
   - 3.2. *ask the requesting agent a_i for the required resource facts RequestSet.*
   - 3.3. **for each** *agent $a_j \in A \setminus \{a_i\}$* **do**
     - 3.3.1. *ask a_j for free res. facts equivalent to RequestSet.*
     - 3.3.2. *add these resource facts to R'.*
   - 3.4. **if** *$R' \supseteq$ RequestSet* **then**
     - 3.4.1. *let $R'' \subseteq R'$ be the cheapest set that satisfies RequestSet.*
     - 3.4.2. *add for each $r \in$ RequestSet the corresponding dependency to R''.*
     - 3.4.3. *remove as much actions as possible from a_i.*
     - 3.4.4. *for each involved agent, update the cost reduction of all requests.*

3 Experimental results

3.1 Research questions and set-up

After a formal analysis of the plan-merging algorithm, two questions remained which we tried to answer by doing experiments.

- We know that the worst-case time complexity is $O(n^2)$. We are also interested in the constant term of the average time complexity, e.g., how long does it take to merge the plans of a whole day for a moderately sized taxi company.
- In the previous section, we claimed that single-agent plans can be further optimized by cooperating with other agents using this algorithm. We would like to verify this claim experimentally.

The tests are performed by merging the plans of taxis in a taxi company. From taxi company Zeevang in Purmerend we received information on rides of about 35 taxis exactly as they were planned in January and February 2002. This information includes a taxi number, (start and end) data and time, number of passengers and the origin and the destination location of each order. From this information we reconstruct the plan for each of the taxis, and try to find improvements over these plans as created by the dispatcher using the plan merging algorithm.

We assume that picking up a passenger along an already planned route has no long-term effects on the plan of a taxi. Furthermore, we assume that in this domain the costs of a plan equals the distance driven by the taxis. Consequently,

Table 1: The standard deviation of the fits in Figure 1 and 2.

Δt	run-time fit stdd.	reduction fit stdd.
3	0.310	12.6
6	0.468	21.0
15	0.517	30.9

our goal is to reduce this distance by exchanging orders among taxis (agents). Therefore, one of the resource facts describes a passenger and its attributes (the destination, their preferred pickup time, etc). The most important resource fact, however, describes the ride of a taxi and models the posibility for passengers to travel from one location to another. This is the only type of resource fact that is exchanged between taxis.

When a customer is transported by taxi A instead of by taxi B the additional distance and time needed by taxi A is estimated using the Euclidean distance and an analysis of all drives as extracted from the data set. An exchange is only allowed if the passenger's estimated arrival time is not increased by more than Δt minutes and the detour length of taxi A is less than the distance reduction of the other. Therefore, in this domain we define resource facts equivalent when their time attributes differ at most Δt minutes.

3.2 Results

First we analyze the running time of plan merging. We expect the worst-case running time to be $O(n^2)$ where n is the number of actions. To test this hypothesis, we run the algorithm with a fixed Δt of 3, 6, and 15 minutes and a fixed day on a number of plans varying from 2 to 35. Each test is performed 20 times on a randomly selected set of taxis. For each run we store the total number of actions of all involved plans and the run time on a 1GHz Pentium processor. The results of these runs can easily be fitted with a quadratic function (about $4 \cdot 10^{-5} n^2$), as can be seen in Figure 1. The standard error of these fits is very small, as can be seen in the first column of Table 1.

Next we are interested in the improvement of the plans. For each run we also store the total distance driven by the taxis, before and after the plan merging algorithm. In Figure 2 the difference between these values is plotted against the number of actions. As can be seen, more relaxed time constraints on the arrival time of the passengers lead to more improvement. Furthermore, the total improvement seems linear in the number of actions. The standard deviation of this fit is shown in the last column of Table 1. The *relative* improvement in drive distance (in percentage) is given in Table 2. The main disadvantage of reducing the distance driven by the taxis is that the passengers need more time to get to their destinations. Table 2 also shows the *increase* in passenger travel time (in percentage).

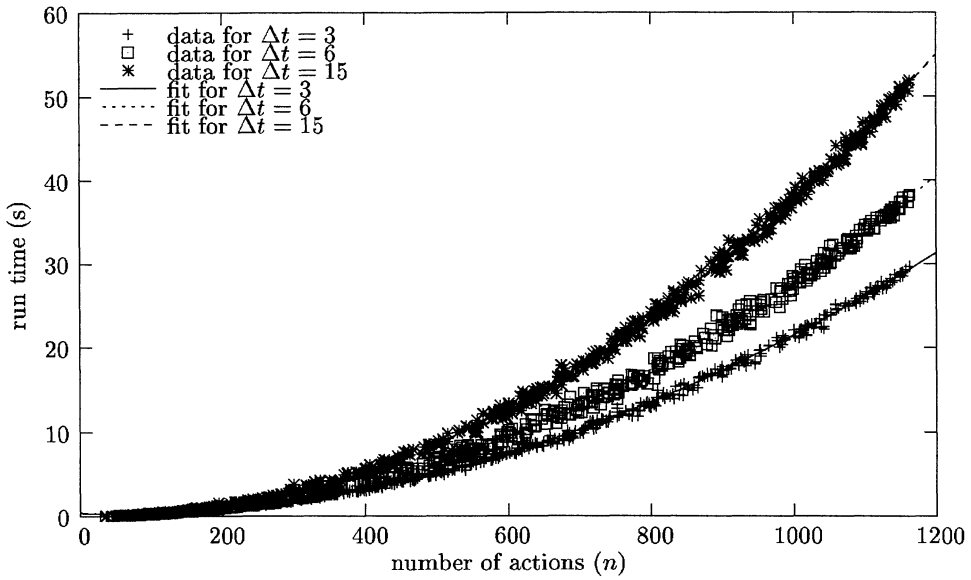


Figure 1: The run time versus the number of actions for three different values for Δt.

Table 2: The standard deviation of the decrease of the drive distance and the increase of the passenger travel time.

Δt	reduction distance (%)	stdd.	increase time (%)	stdd.
3	3.32	2.06	2.60	1.85
6	8.41	4.12	7.05	3.94
15	18.0	7.62	16.4	7.26

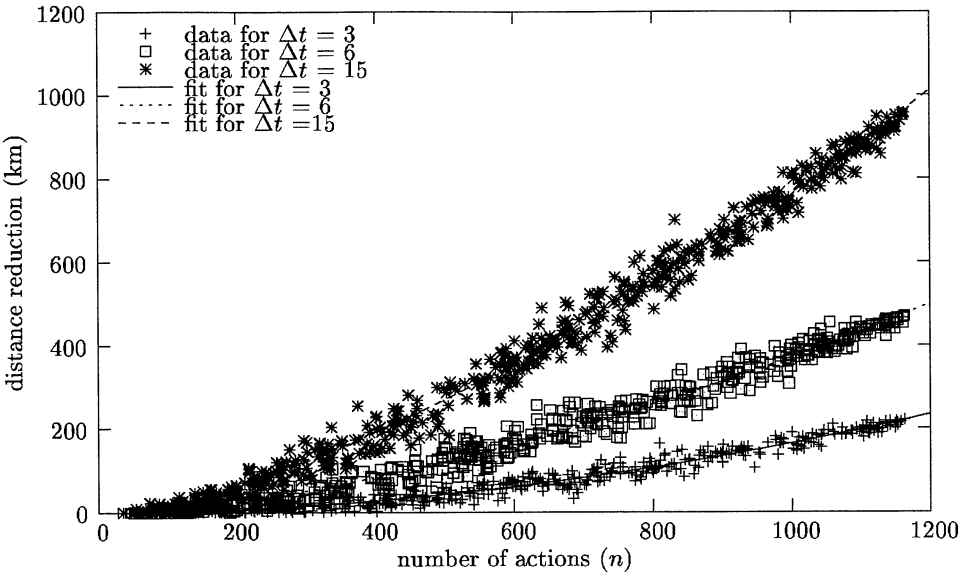


Figure 2: The improvement in drive distance versus the number of actions for three different values of Δt.

4 Discussion and related work

Plan merging is a method for coordinating plans of autonomous agents. For this algorithm we assume that each agent is able to construct a plan for its own problem and we try to obtain a more efficient solution by merging these plans. Experimental evidence is gained by running this algorithm on data from a taxi-company. The results show that this technique is both efficient and effective. The run time is quadratic in the size of all input plans. It takes less than a minute to improve the plan of a whole day of a taxi company consisting of 35 taxis (1200 actions). Within this time the distance driven by the taxis is reduced significantly. When passengers are allowed to arrive within an interval of 3 minutes from their desired arrival time the total distance can be about 5 percent improved over the schedule created by the dispatcher of the taxi company. Moreover, when passengers are allowed to arrive within an interval of 15 minutes, as is the current agreement for most Dutch low-budget elderly transportation services (Vervoer op Maat), the improvement can be up to 30 percent.

Although the plan merging algorithm and the used action resource formalism have already been published [8, 10], in this paper we (finally) show that they are powerful enough to describe and coordinate multi-agent plans. A disadvantage of plan merging is that it cannot be used in situations where an agent needs to cooperate with others to construct a plan. The plan merging scheme can be extended to a *multi-agent planning* method by (i) allowing agents to be able to *request* services from other agents and include the results in their plans, or (ii) by

allowing agents to be able to *offer* services to other agents and, upon a request, add these to their plans. Exchanging services enables agents to not only offer resource facts that are already in its plan (and unused), but also to adapt its plan to produce resource facts that are desired by other agents. Such extensions should lead to a distributed algorithm where self-interested agents create plans including coordinated (efficient and conflict-free) actions.

Most solutions to multi-agent planning problems *(i)* cooperatively create plans for all agents without dealing with the self-interestedness of agents, called cooperative distributed planning [2], such as PGP [1, 3], *(ii)* focus on task allocation [6] and conflict resolution *before* planning [4, 11], or *(iii)* conflict resolution *after* planning [7]. An extension of the plan merging algorithm should integrate coordination and conflict resolution in the planning phase, while maintaining the autonomous and self-interested aspects of agents.

We expect that such a coordinated planning algorithm yields even better results than the current plan merging algorithm, since opportunities to cooperate can be better utilized. Both the basic algorithm and the extensions use a resource fact oriented view on the world, and can be combined with most existing planning techniques. Such a general approach to coordinating plans of multiple agents can be used to solve many practical coordination problems, such as hospital scheduling, coordinating the transportation of goods or people, and managing the planning of joint forces on a mission of the UN.

To be able to use the proposed methods on integrating planning and coordination in these situations, still much work need to be done. Firstly, we need an adequate way to reward agents that offer services and share resource facts. Secondly, we need to know how to deal with agents that cannot or do not fulfill their contracts. Furthermore, we should test the developed algorithms in more realistic environments and improve them with (maybe even domain-dependent) heuristics. In addition, we need to look at a more dynamic (continual) version of the proposed algorithm where planning, replanning and execution are integrated. Finally, such approaches cannot be used in open multi-agent environments (e.g., the Internet) before a way is devised to deal with different ontologies (i.e., what are the resource facts in this domain), and a standard for agent communication and negotiation is chosen.

Acknowledgements

Pim van der Stoel from Tens B.V. and Ron Kooi from Taxi Zeevang made the data test set available.

Mathijs de Weerdt is supported by the Seamless Multimodal Mobility (SMM) research program and Roman van der Krogt and Jonne Zutt are supported by the Freight Transport Automation and Multimodality (FTAM) research program. Both research programs are carried out within the TRAIL research school for Transport, Infrastructure and Logistics.

References

[1] K. S. Decker and V. R. Lesser. Generalizing the partial global planning algorithm. *Int. J. of Intelligent and Cooperative Information Systems*, 1(2):319–346, June 1992.

[2] M. E. DesJardins, E. H. Durfee, C. L. Ortiz, and M. J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 4:13–22, 2000.

[3] E. H. Durfee. Distributed problem solving and planning. In G. Weiß, editor, *A Modern Approach to Distributed Art. Intel.*, chapter 3. The MIT Press, San Francisco, CA, 1999.

[4] D. Foulser, M. Li, and Q. Yang. Theory and algorithms for plan merging. *Art. Intel. J.*, 57(2–3):143–182, 1992.

[5] S. Kambhampati. Refinement planning as a unifying framework for plan synthesis. *AI Magazine*, 18(2):67–97, 1997.

[6] S. Kraus. *Strategic Negotiation in Multi-Agent Environments.* Intelligent Robots and Autonomous Agents. The MIT Press, San Francisco, CA, 2001.

[7] F. von Martial. *Coordinating Plans of Autonomous Agents*, volume 610 of *Lecture Notes on Art. Intel.* Springer Verlag, Berlin, 1992.

[8] J. Tonino, A. Bos, M. M. de Weerdt, and C. Witteveen. Plan coordination by revision in collective agent-based systems. *Art. Intel.*, 142(2):121–145, 2002.

[9] M. M. de Weerdt. *Plan Merging in Multi-Agent Systems.* PhD thesis, Delft Technical University, Delft, The Netherlands, 2003.

[10] M. M. de Weerdt, A. Bos, J. Tonino, and C. Witteveen. A resource logic for multi-agent plan merging. *Annals of Mathematics and Art. Intel., special issue on Computational Logic in Multi-Agent Systems*, 37(1–2):93–130, January 2003.

[11] Q. Yang, D. S. Nau, and J. Hendler. Merging separately generated plans with restricted interactions. *Computational Intel.*, 8(4):648–676, November 1992.

Shortest Solutions for Sokoban

Wieger Wesselink Hans Zantema

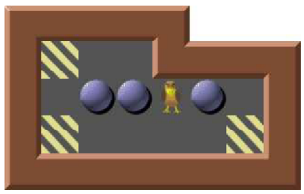
Technische Universiteit Eindhoven, P.O.Box 513 5600MB Eindhoven

Abstract

We describe a way to solve instances of the well-known Sokoban game fully automatically. Instead of developing heuristics in the search for a solution as is usual in this area, by our method we find a guaranteed shortest solution, and even find a representation for all shortest solutions. By using BDD technology we are able to manage state spaces of over 10^{10} states.

1 Introduction

Sokoban is a single player game with very simple rules. Although it can be played as a board game, it became very popular as a computer game round 1980. A board with a rectangular grid contains a number of *stones* that have to be pushed by a *man* towards their goal squares. The man can move to empty neighbor squares and can push a stone if there is an unoccupied square directly behind it. The game is complicated by the appearance of *walls*, that may not be passed. In the picture we give a simple example of a Sokoban problem with 3 stones, drawn as circles. The shaded areas are the goal positions. The man can move down, and it can push a stone to the right. It is not possible to move up due to the presence of a wall, and it is not possible to push to the left since two stones cannot be moved at once. It is very easy to get trapped in a deadlock situation from where a solution can not be reached, for example by pushing a stone into a corner.



It is well known that solving arbitrary Sokoban puzzles is a PSPACE complete problem (see [2]). Usually the Sokoban problem is solved using heuristic search methods like IDA* (see [6]). We will instead solve the problem completely by doing an exhaustive search of the state space. This has the advantage that more powerful results can be found, like a guaranteed shortest solution. On the other hand, this limits the problem size that can be dealt with. To deal with the large state spaces we apply ROBDD's (reduced ordered binary decision diagrams). Thus our approach resembles reachability analysis in model checking, see [1]. Besides that, some optimizations are applied to reduce the state space. These optimizations resemble the ones used in heuristic search methods. As a basis for experiments we have used a set of 90 Sokoban problems that has been used in several other articles, and which is part of the XSokoban program (see [9]). Independently of this paper, Doppelhamer and Lehnert follow a very similar approach to solve

Sokoban problems in the unpublished article [3]. However, they do not consider *push optimal* solutions, and also do not use the *dead spot* optimization.

Section 2 describes how Sokoban can be modelled as a reachability problem. Section 3 summarizes the main properties of BDD's. Section 4 describes the implementation of the problem, Section 5 gives the results and in Section 6 we give some conclusions and directions for further work.

2 Reachability

The game of Sokoban can be modelled as a reachability problem. The general reachability problem is defined as follows. Let $I \subset S$ be a set of initial states, let $F \subset S$ be a set of final states and let $T : S \times S \rightarrow \mathbb{B}$ be a transition relation such that

$$T(x, y) \equiv \text{there exists a transition from } x \text{ to } y.$$

The goal is to determine if there is a path from an element $i \in I$ to an element $f \in F$, by repeatedly applying the transition T to i. We are interested in finding the shortest solution. An algorithm to solve this problem is given by

```
ReachAlgorithm(I, F, T) {
    i := 0
    R_{-1} := ∅
    R_0 := I
    while (R_i ∩ F = ∅ ∧ R_i ≠ R_{i-1}) do {
        Im := {y | ∃x : x ∈ R_i ∧ T(x, y)}
        R_{i+1} := R_i ∪ Im
        i := i + 1
    }
    return (R_i ∩ F ≠ ∅, {R_j}_{j=0...i})
}
```

The algorithm returns true if a solution exists. As a side effect, the algorithm generates a sequence of sets $\{R_i\}_{i=0\ldots k}$, with R_i the set of reachable states after i iterations. When we choose $F = \emptyset$, the algorithm continues until the complete set of reachable states has been found.

For a Sokoban problem, the set I contains one element, which is the initial position of the man and the stones. The final state F contains all board positions in which all stones are on a goal position, and the location of the man is arbitrary. The transition relation T is formed by all valid moves and pushes on the board.

A solution to the reachability problem is a sequence $\{x_i\}_{i=0\ldots k}$ that satisfies $x_0 \in I$, $x_k \in F$ and $T(x_i, x_{i+1})$ for $i = 0 \ldots k-1$. To compute a solution, the sets $\{R_i\}_{i=0\ldots k}$ are needed. The following algorithm computes a solution:

```
PickSolution(F, T, {R_i}_{i=0...k}) {
    choose x_k ∈ F ∩ R_k
    i := k
    while (i > 0) do {
        Src := {x | x ∈ R_{i-1} ∧ T(x, x_i)}
        choose x_{i-1} ∈ Src
        i := i - 1
    }
    return {x_i}_{i=0...k}
}
```

3 Binary decision diagrams

To implement the reachability algorithm from section 2, a data structure representing sets of states is needed that supports the following operations efficiently: intersection, union, equality test, member test and existential quantification. Moreover, the memory requirements for this data structure should be small. We want to be able to represent sets containing more than 10^{10} states, by which hash tables are not suitable. A good candidate for such a data structure are binary decision diagrams (BDD's, see [8]).

A binary decision diagram (BDD) in n boolean variables $\{x_i\}_{i=0\ldots n-1}$ is a rooted, directed acyclic graph, whose

- internal nodes are labelled by a variable x_i and have exactly two outgoing edges, a 0-edge and a 1-edge
- sinks are labelled by the boolean constants 0 and 1.

Each assignment to the variables x_i defines a uniquely determined path from the root of the graph to one of the sinks. The label of the sink can be interpreted as a function value corresponding to the given variables. A BDD can thus represent arbitrary boolean functions or, equivalently, arbitrary subsets of $\mathbb{B}^n$. A BDD is called *ordered* if all paths from the root to a sink respect a given ordering of the variables. A BDD is called *reduced* if it contains no node with identical successors, and if no two nodes u and v contain isomorphic subgraphs. In other words, for a BDD to be reduced, it must share identical subgraphs as much as possible. The class of reduced ordered BDD's (ROBDD's) was introduced by Bryant in 1986 (see [Bryant]). Given a fixed ordering of the variables x_i, it has the following interesting properties:

- ROBDD's form a canonical representation of boolean functions. Thus two functions are equal if and only if they have the same ROBDD.
- Basic operations on ROBDD's like intersection and union can be implemented efficiently.
- For many practical instances of boolean functions, the corresponding ROBDD is quite small.

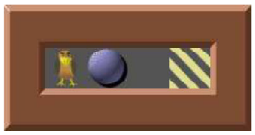
The number of nodes in an ROBDD can sometimes be orders of magnitudes smaller than the number of elements in the subset that it represents. We exploit this feature to represent large sets of Sokoban positions efficiently. In the remainder of this paper when we use the term BDD, we actually refer to an ROBDD.

4 Implementation

In this section we demonstrate how Sokoban can be implemented. For that, both the Sokoban instance and the game rules are encoded in terms of BDD's.

4.1 Binary Encoding

In this section we encode the Sokoban problem in boolean variables, to make it suitable for a representation in terms of BDD's. Only the non-wall squares of the board need to be considered. These squares are numbered from 0 to $n-1$. With each of these squares a boolean variable b_i is associated that describes whether or not the square is occupied by a stone. For the position of the man we introduce $\lceil \log n \rceil$ boolean variables s_j that describe the bit pattern of the number corresponding to the square it is on. Thus we need $p = n + \lceil \log n \rceil$ boolean variables to describe an arbitrary position of a Sokoban problem. In the given simple Sokoban problem containing four squares, numbered from 0 to 3, the given position is described by the boolean formula



$$\neg b_0 \wedge b_1 \wedge \neg b_2 \wedge \neg b_3 \wedge \neg s_0 \wedge \neg s_1,$$

which relates to the initial position I in the reach algorithm. The push from square 0 to 1 is described by the formula

$$(\neg b_0 \wedge b_1 \wedge \neg b_2 \wedge \neg s_0 \wedge \neg s_1) \wedge (\neg b'_0 \wedge \neg b'_1 \wedge b'_2 \wedge \neg s'_0 \wedge s'_1) \wedge (b_3 \leftrightarrow b'_3),$$

where the variables with a prime correspond to the position after the transition. The number of possible pushes in the problem of the picture is four, of which only two can actually be realized from the initial position. The transition relation T of the reach algorithm is a disjunction of the formulas of all possible moves and pushes on the board. It can be built incrementally using basic BDD operations. The final relation F is the disjunction of all goal squares. In the example this is b'_3.

In this way, the reach algorithm from section 2 can be implemented entirely in terms of BDD's. There is one operation that is not immediately obvious in the algorithm. The BDD for the transition relation T is an expression in variables before a transition (x) and after a transition (x'). When existential quantification is used for computing the set Im, the result is an expression in x'. This result has to be converted to an expression in x by renumbering the variables. This is again an efficient operation on BDD's.

Order of the variables The definition of an ROBDD depends on the choice of an order on the variables. This order has an enormous impact on the size of the BDD's. Generally speaking, variables that are closely related (like the variables s_j that describe the position of the man), or appear together in subformulas, should be chosen adjacent. An important observation is that the variables s_j should be chosen smaller than the variables b_i. This can be explained as follows. When $b_i < s_j$ for all i, j, then the subgraphs in terms of s_j variables that describe the position of the man appears at the top of the BDD. Since many of these subgraphs are the same, it is to be expected that the amount of sharing is very high. If the order instead is turned around, these subgraphs appear at the leaves of the BDD. In that case, no sharing is possible at all.

Furthermore, the numbering of the non wall squares (i.e. the order of the b_i variables) is important. We have done some experiments with manually numbering the non-wall squares in such a way that neighboring squares are close. It turned out to be a reduction of about 50% in the maximum size of the BDD's R_i that are produced by the reach algorithm. Modern BDD software packages contain a facility to automatically reorder the variables. In our experiments, it turned out that automatic reordering of the variables produces about the same results. So we decided to simply number the squares from left to right and top to bottom, and apply automatic reordering when necessary. In this way our method is fully automatic: apart from entering the Sokoban instance nothing has to be entered manually to obtain the full analysis of all shortest solutions.

4.2 Optimizations

The size of the state space of the problems in the XSokoban set is huge. A precise estimation for the size of the state space is difficult to give, but a reasonable approximation can be obtained as follows. Let the number of non-wall squares in a Sokoban problem be n and the number of stones be k, then the number of possible positions of k stones and one man is given by $(n-k)\binom{n}{k}$. This is an upper bound for the size of the state space, since many positions can not be reached from the initial position. But in practice this number gives a reasonable estimation of the number of reachable states. For the problems in the XSokoban set this value ranges between 10^9 and 10^{37}.

Dead spots A *dead spot* is a square from which a stone can never reach a goal square, see [5]. The XSokoban problems contain many of such dead spots. Since pushing a stone to a dead spot on square i results in a deadlock, the value of the variable b_i must remain false during the reachability algorithm. A straightforward optimization is therefore to eliminate all variables corresponding to dead spots from the problem by fixing them to false, and to remove all pushes towards dead spots from the transition relation. Finding dead spots is in fact a simplified version of the Sokoban problem itself, but with different initial and final positions. To determine if square i is a dead spot, the initial state I and the final state F can be chosen as

$$I = (\bigwedge_{j \neq i} \neg b_j\ ) \wedge b_i \wedge S \qquad F = \bigvee_{j \in \text{target squares}} b_j,$$

where S is the disjunction of all possible man positions (possibly except i).

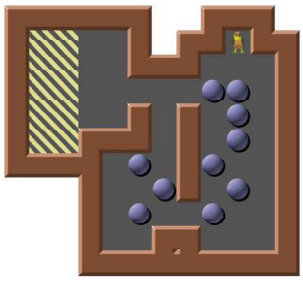
If the final state can not be reached, square i is a dead spot. Since there is only one stone involved, the computation of dead spots is very efficient, and can be done as a preprocessing step. If a Sokoban problem contains m dead spots, the upper bound for the state space reduces from $(n-k)\binom{n}{k}$ to $(n-k)\binom{n-m}{k}$. For the XSokoban problems, this leads to a reduction of the state space with a factor between 10^1 and 10^7.

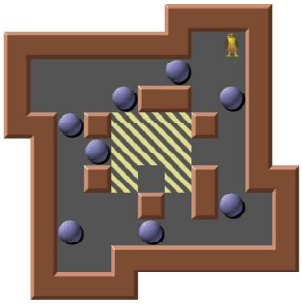
4.3 Push optimal solutions

In practice, two kinds of solutions of a Sokoban problem are of interest: a *push optimal* solution is a solution with a minimal number of pushes and a *move optimal* solution is a solution with a minimal number of man moves (including pushes). These solutions are usually different. Computing a push optimal solution is a reach algorithm, with a more complicated step. Instead of doing one move or one push, the man may do an arbitrary number of moves followed by one push. For doing an arbitrary number of moves again the reach algorithm can be used. The algorithm for computing push optimal solutions is therefore a nested version of the reach algorithm.

5 Results

We have implemented the reach algorithms from the sections 2 and 4.3 in C++, using the BDD software package Buddy, see [7]. For our experiments we used a PC with 512 Mb RAM, and in a few cases 1Gb RAM. First we have done some tests with the Microban levels, a set of small Sokoban problems. Most of them could be solved very quickly. Then we have tried to solve the more serious XSokoban levels. This set has been used as a benchmark for solving Sokoban problems before (see [6]). Without applying any optimization, only the first problem was solvable. With the dead spot optimization we managed to solve the following instances of the benchmark: 1, 6, 17, 38, from which 6 and 38 look as follows:





Even for humans these problems are not straightforward to solve, especially when the shortest solution is wanted. The results are given in table 1.

level	type	# iterations	max bdd size	# reachable states	final bdd size
1	move optimal	230	248541	$4.4 \cdot 10^7$	24829
1	push optimal	97	97061	$4.4 \cdot 10^7$	11750
6	push optimal	110	2134013	$3.8 \cdot 10^{10}$	76694
17	move optimal	503	383975	$1.2 \cdot 10^9$	39919
17	push optimal	213	271265	$1.2 \cdot 10^9$	30071
38	push optimal	81	4298973	$2.6 \cdot 10^9$	195537

Table 1: Results of the experiments with XSokoban problems

In all cases, the solution length matches that of the high scores given on the internet (see [9]). An interesting phenomenon of the reach algorithm is that the BDD size of the sequence $\{R_i\}_{i=0,1,\ldots}$ generated by the reach algorithm often follows the following pattern (see Figure 1): for increasing i the BDD size initially

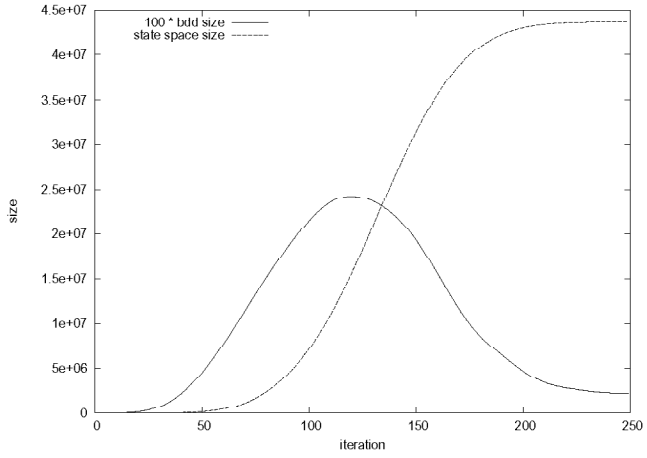


Figure 1: The state space size versus the bdd size of XSokoban problem 1

increases, but at a certain point, when the number of elements becomes large, the size decreases again. This is related to the fact that big well-structured sets can have small BDD's, for instance the BDD representing the complete set (true) has size 1. We observe that often BDD sizes already decrease, while the numbers of states still drastically increase.

The solvability of the reachability problems is closely related to the maximum BDD size during the computation. If this maximum is too big, the memory of the computer may become insufficient. Note that the number or reachable states can become higher than 10^{10}. State spaces of that size would be impossible to represent using hash tables.

The move optimal and push optimal approaches share the initial and final set of the sequence $\{R_i\}_{i=0,1\ldots}$ that is generated by the reach algorithm. The intermediate sets in these sequences are different, and thus the BDD sizes are different too. From our experiments it follows that the maximal BDD size is in general smaller for the push optimal variant of the algorithm. The push optimal problems are therefore easier to solve.

6 Conclusions

Four problems of the XSokoban problem set have been completely solved. The shortest solutions found in high scores on the internet have therefore been proven to be optimal. The sequence $\{R_i\}_{i=0,1\ldots}$ generated by the reach algorithm forms a

rather concise representation of all possible solutions of the corresponding Sokoban problem. If the final set F is chosen empty, the reach algorithm can also be used to compute all positions at maximal distance from the initial position.

The University of Alberta Games Group has spent a considerable effort to solve the 90 problems of the XSokoban set. Using heuristic search methods they managed to solve 57 of them. However, they arrived at this impressive number after applying numerous optimizations. Some of these optimizations are applicable to our approach as well. In [4, 6] so called deadlock patterns are applied, which is a generalization of dead spots . Another optimization follows from the observation that many problems in the XSokoban set can be split into smaller subproblems due to the presence of rooms with a single entrance, see [6].

The authors wish to thank Gia Tagviashvili for his comments and suggestions, and for introducing us to Sokoban.

References

[1] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, 1999.

[2] Joseph Culberson. Sokoban is PSPACE-complete. In L. Pagli E. Lodi and N. Santoro, editors, *Proceedings in Informatics 4, Fun With Algorithms*, pages 65–76. Carleton Scientific, Waterloo, 1999.

[3] Jens Doppelhamer and Johannes K. Lehnert. Optimum solutions for sokoban using obdds. Unpublished, 1998.

[4] Stefan Edelkamp. Planning with pattern databases. In *European Conference on Planning (ECP)*. LNCS, Springer, Toledo, 2001.

[5] Wolfgang Holzinger. Sokoban - facing the infeasible. http://www.logic.at/people/holzi/sokoban/d.ps, 1998.

[6] Andreas Junghanns and Jonathan Schaeffer. Sokoban: Enhancing general single-agent search methods using domain knowledge. *Artificial Intelligence*, 129(1-2):219–251, 2001.

[7] Jorn Lind-Nielsen. Buddy - a binary decision diagram package. http://www.it-c.dk/research/buddy/.

[8] Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI-Design*. Springer-Verlag Berlin Heidelberg, 1998. ISBN 3-540-63869-5.

[9] Andrew Myers. Xsokoban home page. http://www.cs.cornell.edu/andru/xsokoban.html.

Nonmetric Multidimensional Scaling: Neural Networks versus Traditional Techniques

[a]M.C. van Wezel [b]W.A. Kosters

[a] Faculty of Economical Sciences, Department of Computer Science,
Erasmus University
P.O. Box 1738, 3000 DR, Rotterdam, The Netherlands.
E-mail: mvanwezel@few.eur.nl

[b] LIACS, Leiden University
P.O. Box 9512, 2300 RA Leiden, The Netherlands.
E-mail: kosters@liacs.nl

Abstract

In this paper we consider various methods for nonmetric multidimensional scaling. We focus on the nonmetric phase, for which we consider various alternatives: Kruskal's nonmetric phase, Guttman's nonmetric phase, monotone regression by monotone splines, and monotone regression by a monotone neural network. We use sequential quadratic programming to estimate the weights of the neural network. All methods are briefly described. An experimental comparison of the methods is given for various synthetic and real-life datasets.

1 Introduction

This paper concerns visualization of multidimensional data using nonmetric multidimensional scaling. Generally stated, multidimensional scaling (abbreviated MDS) is a collection of techniques for embedding dissimilarity data, given in the form of a dissimilarity matrix, in a space with a chosen dimension. The embedding is often used for the purpose of data visualization and exploratory data analysis. In this sense, MDS is a competitor for other data visualization techniques, such as a Kohonen SOM and principal component analysis. Traditional MDS techniques are subdivided into metric MDS, where the dissimilarities between objects are assumed to be proportional to Euclidean distances, and nonmetric MDS, where the dissimilarities are only assumed to be related to Euclidean distances by some unknown monotone transformation. Nonmetric MDS nevertheless attempts to find a good embedding in Euclidean space by finding the inverse of the transformation. We describe the MDS problem in Section 2. In the case where the dissimilarities represent, e.g., distances between the capitals of the European countries, the Euclidean distance assumption of metric MDS is realistic. However, in the case of dissimilarities between soda brands that have been reported by a panel of test persons, nonmetric MDS seems more appropriate[1].

[1] Merely the data collection method employed in such situations may lead to distortions.

Although traditional MDS is a well established pattern recognition technique (see, e.g., [1, 2]), to our knowledge this subject has not received a lot of attention from researchers in the field of neural networks (or, for that matter, artificial intelligence in general). In particular, all publications concerning neural networks for MDS that are known to us concern metric MDS. In [9] a simple neural network is given for metric MDS. This neural network merely performs a gradient descent on the cost function, which carries the risk of getting stuck in local minima in the error function. To prevent this, in [5] Klöck and Buhmann apply annealing methods from statistical mechanics to the metric MDS problem.

No neural algorithms have been applied to nonmetric MDS so far, besides in [10], where a multilayer perceptron with a special architecture was used to perform the monotone transformation that forms an essential part of all nonmetric MDS algorithms. The aim of the current paper is to extend the method proposed in [10], and to place it in a context by a comparison with other techniques. In particular, in this paper we use the minimization technique sequential quadratic programming for the estimation of the network weights and compare the performance with monotone splines and Kruskal's and Guttman's methods.

This paper is organized as follows. First, an exact problem statement is presented in Section 2. Next, the various methods for the nonmetric phase of MDS are described in Section 3. After that, the results of some experiments are given in Section 4 and finally Section 5 gives conclusions. This paper does not give an in-depth treatment of MDS in general. The reader is referred to [1, 2] for more information on the subject.

2 Multidimensional scaling: problem statement

Before we are able to describe the analysis problem in metric and nonmetric MDS we introduce some terminology and notation:

Dissimilarities: In MDS analyses, the starting point is a matrix $\boldsymbol{\delta}$ of dissimilarities, of which an element δ_{ij} denotes the dissimilarity between two objects i and j. The number of objects is denoted by n.

Embedding: An embedding of the objects in Euclidean space. The coordinates of an object i in this embedding are denoted by $\mathbf{x}_i$. The dimension of the embedding space is denoted by m, so $\mathbf{x}_i = (x_{i1}, \ldots, x_{im})^T$.

Distances: The (real) Euclidean distance between objects i and j is denoted by $\hat{d}_{ij}$. So, $\hat{d}_{ij}$ denotes the distances in the 'true' embedding in m-space. It is this embedding that we attempt to reconstruct. The distance between the estimates for the spatial representations $\mathbf{x}_i$ and $\mathbf{x}_j$ of objects i and j is denoted by $d_{ij} = \| \mathbf{x}_i - \mathbf{x}_j \|$, where $\| \cdot \|$ denotes the Euclidean norm. Collectively the distances are denoted by the matrix $\mathbf{d}$.

Disparities: These quantities are used in nonmetric scaling. Disparities $\hat{\boldsymbol{\delta}}$ are as close as possible to distances between the corresponding coordinate estimates $\mathbf{d}$ but with the restriction that they are monotonically related to the original dissimilarity data $\boldsymbol{\delta}$.

Using the above concepts, the MDS problem can be accurately described. We differentiate between metric and nonmetric MDS. In *metric* MDS, it is assumed that

the dissimilarities are *proportional* to Euclidean distances: $\delta_{ij} = c\hat{d}_{ij}$. One way to obtain a spatial representation of dissimilarity data under the above assumption is to minimize the following error function

$$E_{\text{metric\_mds}} = \sum_{ij,i\neq j} (\delta_{ij} - d_{ij})^2. \tag{1}$$

This minimization gives us the correct coordinates up to a scale factor, a translation and a rotation [2].

In *nonmetric MDS* the 'distance assumption' $\delta_{ij} = c\hat{d}_{ij}$ is relaxed to a 'monotonicity assumption'. It is assumed that the dissimilarities $\boldsymbol{\delta}$ are *monotonically related* to Euclidean distances:

$$\forall i,j,k,\ell : \hat{d}_{ij} < \hat{d}_{k\ell} \Rightarrow \delta_{ij} < \delta_{k\ell} \ . \tag{2}$$

One can look upon the $\boldsymbol{\delta}$ values as being monotonically transformed distance values: $\delta_{ij} = f(\hat{d}_{ij})$ where $f(.)$ is an unknown strict monotonically increasing function. Examples of such functions include some linear, power and logarithmic functions. Nonmetric MDS algorithms estimate a spatial representation for a given dissimilarity matrix in which the rank order of the distances between the embedded objects agrees with the rank order of the dissimilarities as much as possible.

Traditionally, in nonmetric MDS one attempts to minimize the following cost function, often called stress (due to Kruskal [6]), in an iterative fashion:

$$E_{\text{nonmetric\_mds}} = \sqrt{\sum_{ij,i\neq j} (\hat{\delta}_{ij} - d_{ij})^2 / \sum_{ij,i\neq j} d_{ij}^2} \tag{3}$$

Often, the inter-pattern distances are kept normalized ($\sum_{ij,i\neq j} d_{ij}^2 = 1$), in which case the error function reduces to (ignoring the square root)

$$E^{nm} = \sum_{ij,i\neq j} (\hat{\delta}_{ij} - d_{ij})^2, \tag{4}$$

which is computationally simpler. The disparities $\hat{\boldsymbol{\delta}}$ must be re-estimated in each iteration in a so-called nonmetric phase. This nonmetric phase is alternated with a *metric phase* in which a metric MDS problem is solved where the current disparities $\hat{\boldsymbol{\delta}}$ play the role of dissimilarities. The procedure is depicted in Figure 1.

In this paper we do not pay attention to the metric phase of nonmetric MDS — we focus on various possibilities for the nonmetric phase. In the nonmetric phase, the disparities are chosen to resemble the distances $\mathbf{d}$ of the current embedding as close as possible subject to a monotonicity constraint: $\forall i,j,k,\ell : \delta_{ij} < \delta_{k\ell} \Rightarrow \hat{\delta}_{ij} < \hat{\delta}_{k\ell}$. This is done by performing a monotone regression with the current distances $\mathbf{d}$ as targets and dissimilarities $\boldsymbol{\delta}$ as inputs.

By minimizing stress (Equation 3), nonmetric MDS finds a spatial representation in which the dissimilarities are transformed by the monotone transformation in a way that inverts the monotone transformation that distorted the true distances between the objects in m-space, and thus improves the final fit to the data.

[2] Solutions are prone to be local minima. Much research has gone into avoiding this.

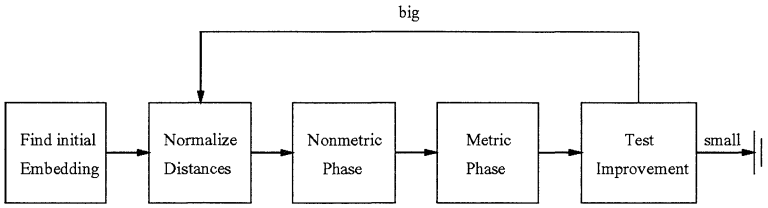


Figure 1: Schematic representation of nonmetric mds algorithm

3 Approaches to the nonmetric phase

Various methods have been proposed to perform the monotone regression in the nonmetric phase. We consider four methods here: Kruskal's nonmetric phase, Guttman's nonmetric phase, monotone regression by monotone splines, and monotone regression by a monotone neural network. Due to space limitations, the presentation here is very brief — for a more elaborate presentation the reader is referred to a forthcoming technical report at `http://www.erim.nl`, or to the references cited in this section. The first two methods manipulate the disparities directly in order to minimize Equation 4, the latter two methods estimate parameters for a monotone regression model which is then used to compute the disparity values for the dissimilarities.

The reason for this explicit modeling using a monotone regression model is threefold. First, explicit modeling gives the user the opportunity to visualize and examine the total monotone transformation, because the regression models perform interpolation. Second, the use of interpolation yields a smooth, continuous mapping from dissimilarities to disparities, which is intuitively more plausible than a step function, which results from Kruskal's and Guttman's nonmetric phases. Third, the use of interpolation in the nonmetric phase can potentially speed up the nonmetric phase considerably in the case where the dissimilarity matrix is of substantial size. The reason for this is that traditional nonmetric phases (like Kruskal's and Guttman's) have a high worst case time complexity. E.g., for Guttman's nonmetric phase it is $\mathcal{O}(n^4)$, whereas for Kruskal's method it is $\mathcal{O}(n^2 \log n^2)$. The time complexity for the monotone regression models is much more favorable. E.g., in the case of the monotone regression splines it is $\mathcal{O}(n^2)$. This leads to a potential benefit for large dissimilarity matrices. Furthermore, the explicit regression models can be 'trained' using only a subset of all available data, and can be used as an interpolator for the remaining data, leading to a further performance gain, but this is not exploited in this paper.

Kruskal's and Guttman's nonmetric phases. In Kruskal's nonmetric phase, disparities are grouped in blocks in which all disparities have equal values. Initially, one block is associated with each disparity in $\hat{\boldsymbol{\delta}}$. The disparity blocks are then compared pairwise in rank order of the associated dissimilarity (i.e., the disparity associated with the i-th largest dissimilarity is the i-th disparity considered), and when a violation of the monotonicity assumption is found, the violating blocks are unified by setting all disparities they contain to the mean of disparities in the two blocks. This is iterated until no violations occur.

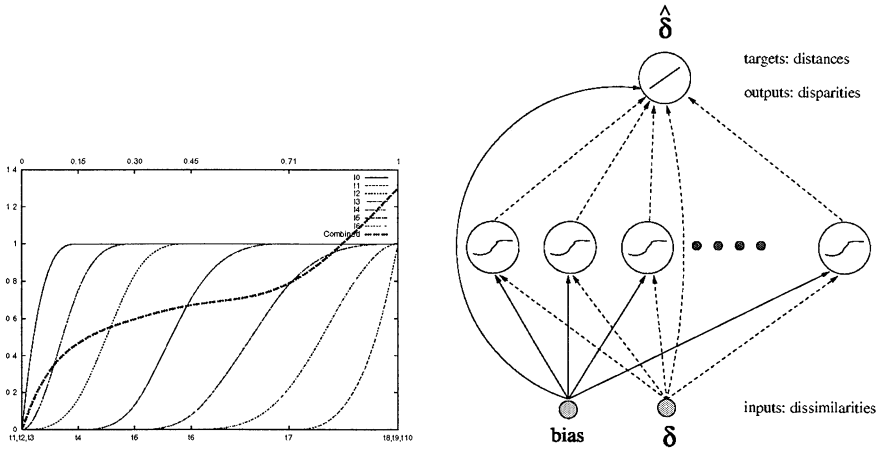


Figure 2: Left: Example of an I-spline basis of order 3 with interior knots situated at $0.15, 0.3, 0.45$, and 0.71, and the linear combination $0.3I_0+0.2I_1+0.1I_2+0.1I_3+0.4I_4+0.2I_5$. Right: Monotone neural network for nonmetric phase in MDS.

In Guttman's nonmetric phase, the disparity associated with the i-th largest dissimilarity is set to the i-th largest element of $\mathbf{d}$ (i.e., the i-th largest distance estimate). Both methods yield a step function as the result and suffer from a high worst case time complexity. In practice the time complexity is hardly a disadvantage because of the small number of objects that is usually embedded.

Nonmetric phase using monotone regression splines. As an alternative to manipulating the disparities directly, one can also construct an explicit model for the relation between dissimilarities and disparities. The first of these models we will consider is monotone regression based on monotone splines, introduced by Jim Ramsay in [8], where the application to nonmetric MDS is also mentioned.

Briefly stated, a monotonically increasing nonlinear function can be approximated by forming a linear combination of a number of *monotone splines* or I-splines, collectively referred to as the spline basis, where the coefficients of the combination are constrained to be positive. An example of a basis of I-splines and a linear combination is given in Figure 2. The number and form of the I-splines in a basis can be controlled by means of some parameters. In our case, estimation of the optimal coefficients boils down to a quadratic programming problem, for which we used the QP solver `BPMPD` by Csaba Meszaros [7].

Nonmetric phase using a monotone neural network. A fourth possibility for the nonmetric phase, is the use of a a multilayer perceptron neural network that is only capable of modeling monotone transformations. We will refer to this network type as a *mono-nn* in the remainder. It takes the dissimilarities $\boldsymbol{\delta}$ as inputs and generates the disparities $\hat{\boldsymbol{\delta}}$ as outputs. It uses the distances $\mathbf{d}$ as targets, and has one hidden layer with non-linear (hyperbolic tangent) transfer functions. The output unit uses the identity as a transfer function. Since the individual transfer functions are monotonically increasing, the monotonicity constraint is always satisfied if we impose a positivity constraint on all weights of the neural

name	description	# objects	reference
`cars`	Pairwise dissimilarity judgments on 11 car models.	11	SPSS
`square`	Nonlinearly transformed distances between points arranged in a square.	16	[10]
`riasec`	Dissimilarities between six vocational preference inventory scales.	6	[2]
`citations`	Transformed inter-journal citation numbers in psychometric literature.	28	[4]

Table 1: Description of used datasets.

network except the biases. A network of this kind is depicted in Figure 2.

We implemented two approaches to enforce the positivity constraint on the weights. Details on these approaches can be found in the forthcoming technical report mentioned earlier. In the first approach, we *squared* all the weight values prior to their use. So, the input to a unit b in the hidden layer would be $in_{b|ij} = w_b^2 \delta_{ij} + \theta_b$, where w_b and θ_b denote unit b's weight and bias. The learning rule has to be altered to incorporate the squared weights. Details can be found in [10] and in the forthcoming technical report.

In the second approach, we used the minimization technique *sequential quadratic programming* (SQP) for estimation of the weights. SQP is a technique for minimization of a general (possibly nonlinear) function under general (possibly nonlinear) equality- and inequality-constraints [3]. It obtains a solution by replacing the original objective function by a succession of quadratic approximations, so that each iteration involves solving a quadratic programming problem. We used the SQP code `CFSQP` in our experiments. `CFSQP` is available for free to the academic research community (see `http://www.aemdesign.com`).

The estimation of the parameters for the mono-nn is accelerated by using a good initial embedding. In the experiments reported in the next section, we used the solution obtained by metric MDS as the initial embedding.

4 Experiments and results

To assess the quality of the final embedding obtained with all nonmetric scaling methods, we used the datasets mentioned in Table 1. Table 2 shows the results of the experiments with these datasets. All results are averaged over 50 independent runs, to compensate for the variance in the final stress-values due to different initial configurations.

One can see that all nonmetric MDS methods clearly improve the data fit compared to the metric MDS method. In general, the 'non-interpolating' nonmetric MDS methods (Kruskal and Guttman) give a better fit than the interpolating MDS methods. The remaining nonmetric MDS methods, the monotone neural network and the monotone spline, are close to one-another with respect to the normalised stress value.

The monotone transformations yielded for the `citations` dataset by the various scaling methods are depicted in Figure 3. It is clearly visible that the non-interpolating methods yield a non-smooth transformation, which is not very plau-

[3] Our constraints are linear while our objective function is nonlinear. Algorithms for this special case also exist (see [3]) but an implementation of such a method was not available to us.

	avg	stdv	min	max	avg	stdv	min	max
	cars				square			
Metric	0.009911	0.000213	0.009845	0.010640	0.017536	0.013033	0.010766	0.043688
Kruskal	0.001215	0.000000	0.001214	0.001215	0.001463	0.007313	0.000000	0.037328
Guttman	0.002609	0.000004	0.002604	0.002624	0.000000	0.000000	0.000000	0.000000
Spline	0.003606	0.000001	0.003606	0.003607	0.023922	0.068711	0.010766	0.500000
Neural	0.007260	0.000201	0.007094	0.008635	0.012375	0.014460	0.002173	0.035308
SQP neural	0.006416	0.001096	0.002187	0.006780	0.006325	0.010753	0.000070	0.035434
	riasec				citations			
Metric	0.023089	0.006914	0.016013	0.036849	0.661513	0.003918	0.658505	0.687850
Kruskal	0.004635	0.006764	0.000000	0.017992	0.057253	0.000293	0.057174	0.059301
Guttman	0.000000	0.000000	0.000000	0.000000	0.072910	0.000031	0.072838	0.072994
Spline	0.170304	0.225297	0.000472	0.500000	0.056720	0.000013	0.056691	0.056744
Neural	0.010163	0.009910	0.000150	0.023258	0.059584	0.001811	0.056677	0.064066
SQP Neural	0.014982	0.010172	0.000072	0.035448	0.057731	0.002347	0.053119	0.064688

Table 2: Normalised stress (E^{nm}) values obtained for all four datasets with metric and various types of nonmetric scaling, averaged over 50 runs.

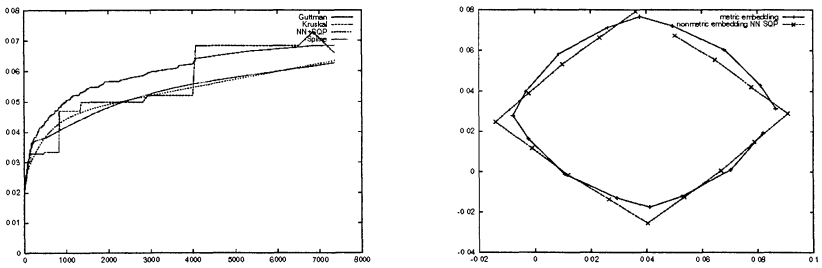


Figure 3: Transformations yielded by the various nonmetric phases for the `citations` dataset (left), and embeddings yielded for the `square` dataset (right) by metric and nonmetric scaling.

sible. Also shown are the embeddings yielded by metric scaling and by nonmetric scaling using a neural network (trained with SQP) in the nonmetric phase for the `square` dataset. It is clear that nonmetric scaling improves the embedding yielded by metric scaling.

To examine the computational demands of the various methods we created artificial dissimilarity matrices of various sizes by randomly generating a number of points n in a 2-dimensional space, calculating the distance matrix and applying a nonlinear monotone transformation ($f(x) = \log(x + 0.01) + 5$) to these distances. Table 3 shows the wall clock times in seconds needed for the nonmetric embedding of these artificial datasets for various n. For large n, the use of a spline method in the nonmetric phase is relatively fast, due to the inherit simplicity of this method. It is clear that the monotone neural network, when trained with the modified backpropagation procedure of [10] is unacceptably slow, but when the SQP method is used, the required computation times are acceptable, whereas the results reported in Table 2 indicate that the obtained stress values are usually lower.

5 Summary and conclusions

We have applied neural networks in the nonmetric phase in nonmetric multidimensional scaling. The weights of the neural network were estimated by an altered

	Number of objects n					
	10	50	100	300	500	t_{500}/t_{10}
Metric	0.01 (.019)	0.21 (.027)	2.03 (.027)	9.99 (.024)	34.01 (.020)	3401
Kruskal	0.28 (.000)	8.81 (.000)	28.83 (.000)	113.87 (.000)	193.55 (.000)	689
Guttman	0.11 (.000)	0.39 (.011)	21.00 (.000)	66.79 (.000)	209.10 (.000)	1900
Spline	2.33 (.000	1.54 (.000)	3.86 (.000)	35.20 (.000)	84.11 (.000)	36
Neural	19.36 (.000)	79.64 (.003)	502.38 (.001)	5051.14 (.001)	89566.09 (.001)	4626
SQP Neural	2.95 (.000)	31.73 (.000)	75.90 (.003)	644.76 (.004)	615.80 (.006)	208

Table 3: Wall clock computation times required by the various nonmetric MDS methods. Shown between brackets are the resulting stress values. The final column shows the quotient of the required time with 500 (t_{500}) and 10 (t_{10}) objects.

training procedure and by sequential quadratic programming. The approach was compared with other approaches (Kruskal's and Guttman's methods and monotone splines) in a series of experiments. The neural network based methods perform comparable to the existing methods, but have the advantage of yielding smooth mappings instead of step functions, which is more plausible and makes interpolation easier.

References

[1] I. Borg and P. Groenen. *Modern multidimensional scaling. Theory and applications.* Springer-Verlag, New York, 1997.

[2] M. L. Davison. *Multidimensional Scaling.* Wiley, New York, 1983.

[3] R. Fletcher. *Practical methods of optimization.* Wiley, 2nd edition, 2000.

[4] P.F.J. Groenen and W.J. Heiser. The tunneling method for global optimization in multidimensional scaling. *Psychometrika*, 61(3):529–550, 1996.

[5] H. Klöck and J.M. Buhmann. Data visualization by multidimensional scaling: A deterministic annealing approach. *Pattern Recognition*, 33:651–669, 1999.

[6] J. B. Kruskal. Multidimensional scaling by optimizing goodness-of-fit to a nonmetric hypothesis. *Psychometrika*, 29:1–29, 1964.

[7] Cs. Mészáros. The BPMPD interior point solver for convex quadratic problems. Technical Report WP 98-8, Hungarian Academy of Sciences, Budapest, 1998.

[8] J.O. Ramsay. Monotone regression splines in action. *Statistical science*, 3(4):425–441, 1988.

[9] M. C. van Wezel, J. N. Kok, and W. A. Kosters. Two neural network methods for multidimensional scaling. In *European Symposium on Artificial Neural Networks (ESANN'97)*, pages 97–102, Brussels, 1997. D facto.

[10] M.C. van Wezel, W.A. Kosters, P. van der Putten, and J.N. Kok. Nonmetric multidimensional scaling with neural networks. *Lecture Notes in Computer Science*, 2189:145–156, 2001.

Generating Powertraces Using Neural Networks

Wouter Th. Wiersma

Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
W.T.Wiersma@stud.tue.nl

Abstract

This paper presents the results of our investigation into the application of neural networks for "predicting" power consumption traces of smartcards. We will show the merit of such an approach, along with a detailed description of the experiments. On the basis of the results of the experiments we are able to make recommendations on the feasibility and desirability of synthesizing powertraces by means of neural networks.

1 Introduction

Security of smartcard implementations has gained momentum due to the publication of power analysis attacks [1, 4]. Power analysis attacks — *simple power analysis* (SPA) as well as *differential power analysis* (DPA) and variations — work by running an algorithm on a smartcard and measuring the power consumption of the card. Differences in data will yield tiny differences in the powertrace of the execution of the program. Analyzing these powertraces, rather the differences between traces, may yield sufficient information for a smartcard designer to test the smartcard against attacks to obtain, for instance, the secret key used by the cryptographic algorithm. One of the problems of this approach is the need to program the smartcard and physically measure the power output. These issues can make a DPA attack a costly and time-consuming enterprise, but by simulating a smartcard in software these problems can be avoided. One of the goals of the PINPAS project [6] is to do just that: simulation of smartcards in software, thus performing power analysis attacks on these smartcards without actually having to program the cards and measuring the power output.

The success of a differential power analysis attack depends on the amount of information within the powertraces. For this reason the PINPAS-tool has to interpret a program given for a particular chipset and generate a powertrace related to the execution of the program. In general, the more detailed the synthesized powertraces, the more information can be derived from these traces, and the better a DPA attack can be deployed. Herein lies one of the major challenges of the PINPAS project: synthesizing powertraces that provide the same amount of information as physically measured powertraces. With these results the security of smartcards can be evaluated at an earlier stage of the design, which can save a lot of time, effort, and money.

At present the PINPAS tool generates powertraces by taking the Hamming weight of the data sent over the memory bus. Although this model of powertraces seems far too simple compared to the actual power output of a smartcard, the results obtained with this model are fairly adequate. So far, successful DPA attacks have been applied against TEA, DES, and AES using the PINPAS-tool. However, a more sophisticated power model will enable more powerful attack scenarios as described in [2].

Powertraces that depend solely on the Hamming weight of bus traffic is still a pretty realistic approach for modeling smartcard processors of which the architecture is not very complex and the design of the processor does not take into account the possibility of a power analysis attack. In other words, this powermodel works fine for processors designed before 1996. Modern processors, however, have built-in countermeasures and are generally more complex, which, consequently, causes the analysis of these powertraces to be far more difficult. The designers/manufacturers of these processors undoubtedly already have far more advanced powermodels for these processors, but are understandably apprehensive in disclosing this insider knowledge to the public. Given that the powermodel of these modern processors has some underlying logic, we try to approximate the behavior of this powermodel. The subject of this paper is to approximate this behavior of modern processors by training neural networks to yield realistic powertraces.

Section 2 describes the theoretical background of power consumption traces concerning this investigation. Using neural networks (see [5] for detailed information) we will perform some experiments in order to show the capabilities of these networks in generating powertraces. Section 3 will give a detailed account of the set-up for the conducted experiments. In the subsequent section the results of the experiments will be presented along with an interpretation of this data.

2 Power Consumption Traces

In order for a neural network to be useful in generating powertraces there has to be a functional dependency between the program instructions with their operands and the associated powertrace. The powertraces of different instructions often appear non-related — different instructions perform different internal operations and the execution time is usually also different — although common parts of instructions can be present (e.g., an addition and a multiplication both read values from memory). But the same instruction with different values for the operands is a different matter. Although the operands for an instruction may have a large range, the general shape of the traces is roughly the same. From a practical point of view it would seem wise to have a different neural network for each individual instruction of the processor to be modeled, instead of one network for all instructions. Each of these networks has to be trained with a training set with input consisting of, at least, the value of the operands, and output consisting of the corresponding powertrace.

As mentioned before, modern-day processors are extremely complex which causes additional problems. One major problem is that the powertrace for an

instruction does not only depend on that one instruction and its operands, but also on previous instructions. For this to be properly trained a "history" of the instruction also has to be taken into account as input for the network. This causes the size of the training sets to grow dramatically. The trouble with training neural networks for powertraces is that the training set has to include realistic powertraces, which have to be measured physically, or adequate representations thereof. Obtaining the real powertraces is a very time-consuming business, so the smaller the required training set to correctly generate other powertraces, the better. If a powertrace has to be measured for every possible history of an instruction as well as for every operand value this could quickly become unfeasible. On the other hand, the effect previous instructions have on the powertrace of an instruction might be the same for each instruction. Also the fact that some combinations of adjacent instructions never occur decreases the amount of possible history that has to be considered. At this time we don't exactly know what the effect of previous instructions is on the powertraces, so we have not taken this into account during our experiments.

3 Setting up the experiments

Since the size of the training set is the bottleneck in the process of training a neural network to generate powertraces, we want empirical data to show the trade-off between the size of the training set and the error between the synthesized and the measured powertraces.

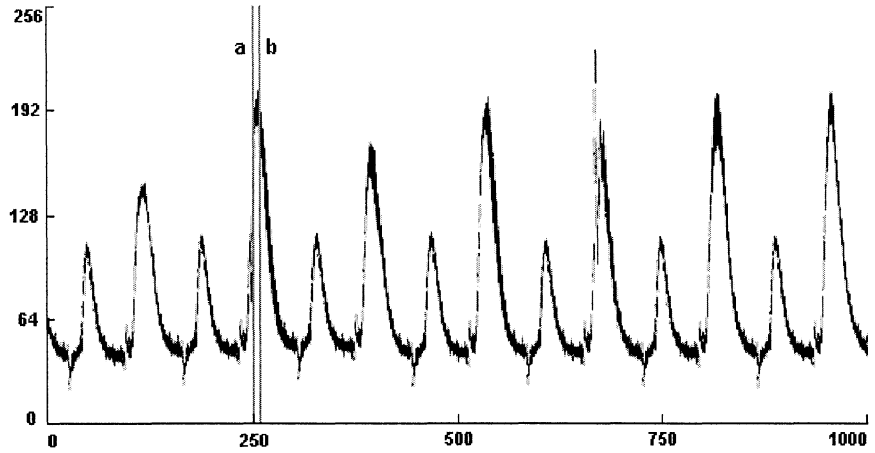


Figure 1: 256 powertraces plotted on top of each other

For these experiments we will consider the powertraces over all 256 possible operand values of one instruction, all with the same preceding instruction. The smartcard we used is the Atmel AT90S4414, which is an easily programmable 8-bit RISC microcontroller. The 256 powertraces are the result of measuring the

assignment of all 8-bit values to a fixed memory-address. These measurements were performed at TNO TPD (Delft, The Netherlands) with a sample frequency of 500 MHz; this amounts to powertraces consisting of 1000 data points (the current amplitude at each time), which are integer values between 0 and 255. Figure 1 shows all these 256 powertraces on top of each other. This graph shows where the powertraces depend on the data; these are the parts where the graph is thickest.

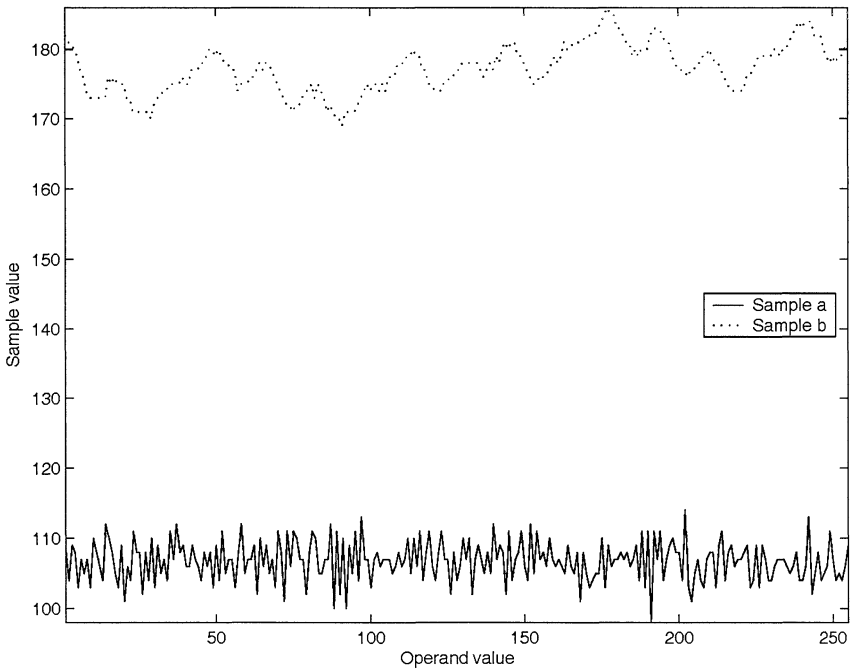


Figure 2: Value of two powertrace samples for all 256 operand values

Trying to learn a neural network all of these 1000 points for every input is a very arduous task, so we will look at only two representative points (indicated in Figure 1 as sample a and b). We conjecture that if experiments show that the network can successfully learn these two values, then given a larger network and more learning time, all the 1000 sample points can be learned satisfactorily. Figure 2 shows the value of these points for all 256 operand values; these values are the target output of the neural network. Figure 2 clearly shows the non-linearity between the operand value and the sample value, which means that trying to learn these values given a single input might be too difficult for the network to handle. For this reason we will represent the operand value not as a single integer in the range [0..255], but as eight separate values in the range (0..1) corresponding with that value's binary representation.

So, the patterns in the training set consist of the bit representation of the operand value and the corresponding sample value of both samples a and b. For

reasons of convenience we will scale the output from [0..255] to (0..1). This way we can use the standard activation function ($f(x) = 1/(1 + e^{-x})$) for all layers of the network. The input of the network consists at this time of just eight values, the bit representation of the operand value for the instruction. The number of inputs could be increased to allow the (partial) powertrace of the preceding instruction(s) to be considered. The number and the size of the hidden layers will be fixed for these experiments which will be performed using two different learning rules: the standard error backpropagation (incremental error backpropagation), and the batch version. The measurement of the error between the desired output and the actual output of the network will be the mean squared error (MSE).

We wanted to perform the experiments in an widely available neural network simulator, and decided on the Stuttgart Neural Network Simulator [3] (SNNS). This toolkit allows for rapid construction of neural networks and easy training sessions. The bonus of this package is that it can export a neural network to a C program file; this allows a trained neural network to be embedded in another application (for instance, the PINPAS-tool).

4 Results of the experiments

In the experiments we will train a standard feedforward neural network with a fixed topology and a variable training set. The topology of the network will be of the form:

- one input layer consisting of eight units;
- one output layer consisting of two units;
- two hidden layers consisting of 100 units.

Adjacent layers are fully connected, which means the network consists of 210 units and 11000 connections.

Figure 3 plots the error associated with a certain training set size. The vertical axis describes the mean squared error, while the horizontal axis describes the fraction of the entire training domain that is used as training set. For this particular task it might be more appropriate to customize the topology in order to increase performance. The same can be said for the learning rules used; other (more advanced) learning rules might reduce the size of the training set, while maintaining the same degree of success.

Figure 3 shows what was to be expected: the larger the training set, the better the network performs. What is kind of surprising is that the batch version of the learning rule performs significantly better than the incremental version, and, therefore, is to be preferred for this task. The time it took to train the network using the batch version was, however, a lot more than the other algorithm. So, if training time is also important, the choice for the standard error backpropagation learning rule might be made.

Please note that the output values are in the domain (0..1) and MSE squares the error. For example, a MSE of 1.0×10^{-4} means that the average error over

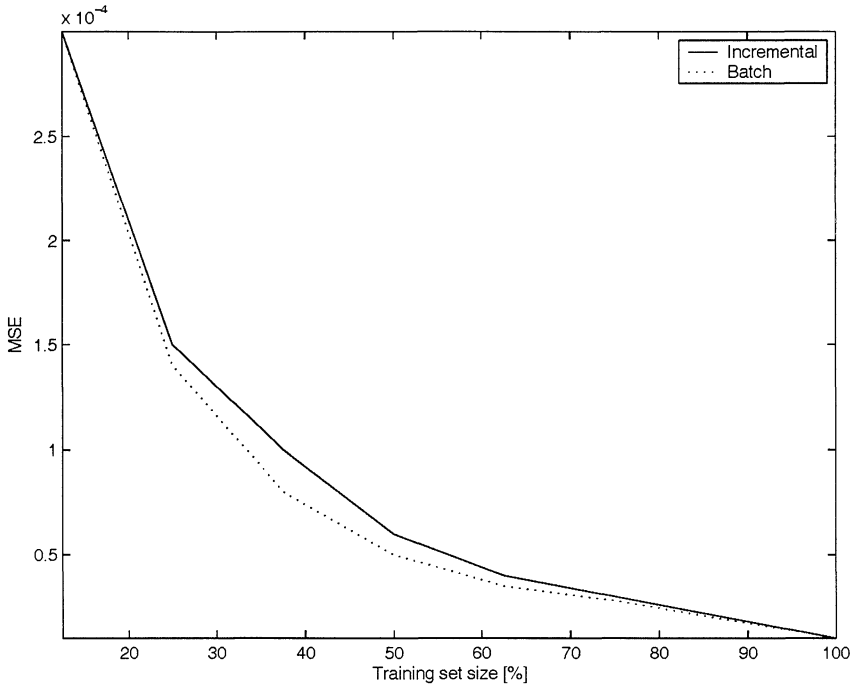


Figure 3: Fixed topology, variable training set

the two output values is 1.0×10^{-2}. After scaling, this is a considerable deviation of 2.56 averaged over both sample values. Given the small margin of error allowed by the data in Figure 2 this might cause some problems. It is up to the user of the data to decide what level of error is allowed, and if that level can be accomplished by a feasible training set.

5 Conclusions and recommendations

The purpose of this paper was to investigate the use of neural networks for generating powertraces. By describing the possible use of such an approach and applying the theory to a practical example we have shown that neural networks can indeed be used to synthesize powertraces. Although the experiments were quite successful, the viability of this approach for specific applications remains to be seen. In other words, whether neural networks can be used for generating powertraces depends on the user of the traces. Questions like "What level of error is allowed?", "Can the required amount of training patterns be provided in reasonable time?", and "How much does the training of the network cost?" are crucial and have to be answered for every application of this approach.

At this time we are skeptical of the success of synthesizing powertraces using

neural networks. We feel that the use of neural networks to synthesize powertraces requires a very large training set, perhaps too large to be practical. Considering the time it takes to physically obtain powertraces as training patterns and the influence of the history of an instruction, it might take too much time to greatly benefit from this approach. However, the use of especially designed topologies (e.g., some sample points are constant, regardless of the operand value, and do not require a connection to the input layer) might significantly improve the performance of the neural network approach. One could also think of dedicated learning rules, instead of the general approach demonstrated in this paper, in order to reduce the size of the required training set.

6 Acknowledgements

The author would like to thank the people at the EIB department of TNO TPD for providing the measured powertraces. We also like to thank Huub ten Eikelder for all his insightful comments on neural networks.

References

[1] Paul Kocher, Joshua Jaffe, Benjamin Jun. Differential power analysis. *Proceedings of Advances in Cryptology — CRYPTO'99*, pages 388–397, 1999.

[2] Mehdi-Laurent Akkar, Régis Bevan, Paul Dischamp, Didier Moyart. Power Analysis, What Is Now Possible.... *ASIACRYPT 2000*, pages 489–502, 2000.

[3] A. Zell, G. Mamier, et al. *SNNS Stuttgart Neural Network Simulator, User Manual, Version 4.2.* available at:
`ftp://ftp.informatik.uni-tuebingen.de`.

[4] T.S. Messerges. *Power Analysis Attacks and Countermeasures for Cryptographic Algorithms.* PhD thesis, University of Illinois, Chicago, 2000.

[5] D.W. Patterson. *Artificial Neural Networks.* Prentice Hall, New York, 1996.

[6] Jerry den Hartog, Jan Verschuren, Erik de Vink, Jan de Vos, Wouter Wiersma. PINPAS: A Tool for Power Analysis of Smartcards. *Proceedings SEC 2003*, pages 453–457, 2003.

Information Markets for Agent-Based Retrieval

Floris Wiesman Geert Graat Evgueni Smirnov

IKAT, Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht

Abstract

In contrast to Internet search engines, search agents deliver up-to-date documents for specific users. The article investigates the improvement of search agent performance through the exchange of documents between search agents. Documents are exchanged on an information market, where search agents can meet and trade documents. Two information market-architectures are presented: the free-exchange architecture, where search agents exchange all documents, and the paid-exchange architecture, where search agents only exchange documents when they can afford it. The information market architectures are compared to a collection of search agents that do not collaborate at all. Experiments showed improved effectiveness for agents that used an information market.

1 Introduction

Current Internet search engines perform remarkably well. Search agents can even perform better, because they do not depend solely on the never-up-to-date databases of search engines, but also can search the Internet directly by traversing links. Search agent performance could even be higher when agents would be able to exchange documents. In this paper we propose the use of a market where search agents meet and exchange information. We investigate architectures with two kinds of markets and compare their effectiveness and efficiency with a basic search agent architecture.

Related to our approach are HARVEST, CARROT II, I-GAIA, and JXTA SEARCH. HARVEST [1] was one of the first architectures that dealt with distributed information retrieval (IR). Information from various sources is aggregated on several levels and made accessible by a hierarchy. CARROT II [3] is based on the exchange of information between agents that each represent a collection. The exchange can be performed in three ways: (i) from each agent to all other agents, (ii) from each agent to one broker, who routes the information to the appropriate agents, and (iii) amongst all group leaders and within a group, where group leaders are elected representatives of a group of agents. I-GAIA [4] is based on a ecologically inspired multi-agent platform and has many kinds of agents. Information is routed by special agents from search agents to information-offering agents, and vice versa. JXTA SEARCH [5] is a peer-to-peer information-sharing architecture. Peers can offer information and query for information; queries are routed to information-offering peers via hubs. As opposed to our approach none of these four systems uses markets to exchange information.

Section 2 describes our information markets, Section 3 proposes an experimental platform for comparing information markets, Section 4 reports on the experiments that were carried out, and Section 5 provides conclusions and directions for future research.

2 Information markets

[2] have shown that the Internet is too large and complex to be searched comprehensively. Search agents therefore cannot be expected to perform perfectly. Hence, techniques are required that provide ways of searching other than current search engines. To do so, we introduce an *information market*, where a search agent, after it has searched the Internet, may look for additional relevant documents. Search agents also may submit documents they found on the Internet to the market. How the storage of documents on the information market is organized depends on the information-market architecture. Below we describe an architecture where no documents are exchanged: the no-exchange architecture. It serves as a basis for comparison with information-market architectures. Subsequently we describe two such architectures: the free-exchange architecture and the paid-exchange architecture.

The no-exchange architecture

The task of a search agent in any of the three architectures, including the no-exchange architecture, is to retrieve all relevant documents on the Internet, based on a query posed by a user, and to present the results to the user. The search agent in the no-exchange architecture does not exchange documents with other agents.

The *lifecycle* of a search agent is defined as the possible states of the agent during the time span in which it tries to complete its task. The lifecycle of the search agent in the no-exchange architecture comprises the following states. A user assigns a query to the search agent. The agent then starts searching on the Internet for documents relevant to the query, for a fixed duration. Next, it orders the documents found according to their relevance with respect to the query. Finally, the ordered documents are presented to the user. Once the search agent has presented the documents, it becomes free to receive another query, and start its lifecycle again.

The free-exchange architecture

The first information-market architecture is an extension of the no-exchange architecture. The lifecycle of an agent in the free-exchange architecture is for a great deal the same as that of the agent in the no-exchange architecture. The difference is that this agent sends its stored documents to an information market for the benefit of other agents.

The free-exchange information market consists of a number of booths. Each booth corresponds to a query that is being processed by an agent. It is therefore called a *query place*. When an agent receives a query from a user, it opens a

query place on the information market. The idea of the query place is that other agents can contribute documents they found on the Internet to the query place. It consists of a list of documents ranked by their relevance to the query from the agent that created the query place. Documents are only contributed to a query place if they are relevant and not yet present. The agent proceeds with searching the Internet, in the same manner as in the no-exchange architecture. Once the searching is stopped after a particular duration, the agent has found a number of documents, sorted by their relevance to the query, from which it forms a list. The agent then checks if other agents contributed documents to its query place. If that is the case, the agent retrieves the list from the query place and merges it with the list of documents it found on the Internet. The query place is then removed from the information market. All the documents of the merged list are contributed to all the query places on the information market, so other agents can benefit from the documents the agent has found. Finally the merged list is returned to the user. Since the exchange of documents in this architecture is free, this architecture is called the free-exchange architecture.

The paid-exchange architecture

The paid-exchange architecture aims at an information market that uses less space and is more efficient than the free-exchange architecture. It uses the same query places, but their function is different: now agents have money with which they can buy and sell documents. These transactions happen at the query place. Agents have to pay for offering documents on an query place and they can buy documents from their own query place, which corresponds to their query. The agents receive money for every query they process for a user. By limiting the amount of money agents have, the space used by the information market can be controlled. Since the exchange of documents is no longer free but costs money, this architecture is called the paid-exchange architecture.

The lifecycle for an agent in the paid-exchange architecture resembles that of an agent in the free-exchange architecture. A user produces a query that is assigned to an agent. The agent opens a query place, and then starts searching the Internet. After the agent has retrieved the documents it found on the Internet, the agent can decide to buy additional documents from its query place. That is, if other agents decided to put documents on the query place for selling and the agent has sufficient money to buy them. If an agent buys a document, it pays the agent that offered the document. Once the agent has bought all the documents or is out of money, it merges the documents bought with the documents found on the Internet. The agent can now decide to sell its own documents to other query places. Since offering documents costs money and agents have a limited amount of money, the agent will only offer documents on query places for which it has relevant documents. Moreover, when the agent has many relevant documents for some query place, it will offer the most relevant ones because they are most likely to be sold (and therefore earn the agent money). Once the agent has offered the documents, it submits the merged list to the user. Then the agent can continue with another query. The offered documents may be bought by another agent, which will then pay the agent, allowing it to spend more on buying or selling other

documents.

The advantage of this approach is that the space used by the information market is reduced and that the remaining space is used more efficiently, since agents are encouraged to consider at which query place they will offer their documents, because they need the money they receive when the documents are sold to buy their own documents. The free-exchange market lacks this incentive, which leads to flooding the information market by sending the documents to all query places. Another advantage is that the system is robust to malicious agents. A malicious agent can flood the free-exchange market with useless documents, but when it does on the paid-exchange market, the documents will not be bought by other agents since they are useless to them. Hence the malicious agent will not receive any money and therefore cannot offer more documents on the market.

Measures for evaluation

The evaluation of the effectiveness of IR systems is based on two measures: *recall* and *precision*. Recall is the part of the relevant documents that was actually found, and precision is the part of the found documents that was actually relevant. The goal for traditional IR systems is to make a distinction between relevant and not relevant documents, so as to achieve a high recall as well as a high precision. While achieving a high recall may be trivial in a traditional IR system (at the cost of low precision, by retrieving *all* documents), it is not in a distributed IR system where retrieving all documents is impossible. In our research we concentrate on finding all relevant documents, rather than on a making a good distinction between relevant and non-relevant documents. Hence we assume that users and search agents have the same distinction between relevant and non-relevant documents, which results in a precision of 1.0 for all queries. This makes the comparison of information markets easier because we only need to take the recall into account.

Since an information market aims to enlarge the set of relevant documents found by a search agent, the recall of search agents that use the market should be higher than the recall of search agents that do not use the market. How much higher it is depends on the *effectiveness of the information market*. To assess this effectiveness, we introduce the *contribution* of an information market. The contribution is defined as the average recall of a search agent in an information-market architecture divided by the average recall of a search agent in the no-exchange architecture.

To measure the efficiency, we define the *market load*, which is an indication of the size of the information market. The market load is measured by the number of documents on the market at specified time intervals. This makes it possible to determine the average and maximum market load. In terms of efficiency, a low market load is preferred over a high market load, because the market load is an indication of the size of an information market.

To measure the efficiency when the number of search agents increases we introduce a second efficiency measure, the *expansion* of an information market. The expansion is the increase in market load per agent. With the expansion, we can predict the development of the efficiency of an information market when adding more agents. The expansion also depends on the size of the Internet.

3 Experimental platform

The three architectures decribed in Section 2 were implemented in an experimental platform, called IRIS. It consists of four agent types:

The Internet Agent This agent simulates the Internet so as to make the experiments repeatable. It possesses the two characteristics of the Internet that are important for our experiments. (1) The time to retrieve a document on the Internet varies; therefore in IRIS this time is randomized. (2) Hyperlinks and search engine databases change frequently; therefore in IRIS the results of a search are randomized. Documents are represented as tf.idf vectors, which is also the case for the other agent types.

Query Generator Agent This agent simulates a user. It supplies search agents with queries and evaluates their effectiveness. Queries are represented as tf.idf vectors; a document is relevant to a query when the similarity between the two vectors exceeds a threshold.

The Market Agent This agent handles requests from search agents for the exchange of documents and determines the efficiency of the information market.

The Search Agents These agents have only one task: to search. For this task they communicate with the Internet Agent. While the Search Agents search, they may interact with the Market Agent. The current IRIS implementation uses fixed prices and has a straightforward strategy for the search agents on the payed exchange: they use half of their money to buy the top-ranked documents in their query place, and they use half of their money to sell document to other query places. The higher the similarity between a search agent's query and the query of a query place, the more documents the agent will offer there.

4 Experiments

We conducted experiments with the no-exchange architecture, the free-exchange architecture, and the paid-exchange architecture. Experiments conducted with the latter two are also referred to as experiments with the information markets.

An experiment consisted of a run from IRIS with a number of search agents belonging to one of the architectures. Each experiment was repeated ten times. While the search agents searched, their effectiveness was measured. When the search agents in an experiment belonged to an information-market architecture, the efficiency of the information market was also measured.

During the experiments the duration that one search agent searches the Internet per query and the total duration of an experiment were kept the same. The documents and queries were taken from the Ohsumed test collection; we selected the 1991 subset consisting of 73,488 documents and 106 queries. We used these documents to simulate the Internet. For this purpose, the documents were divided into document sets of different sizes. Each document set was filled with randomly chosen documents.

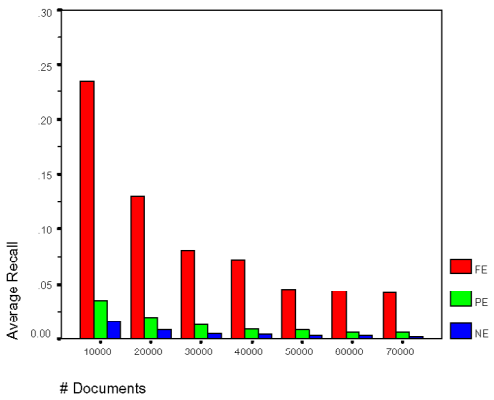


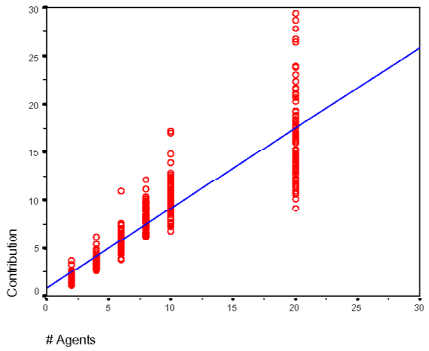
Figure 1: The average recall of a search agent in an experiment with 20 search agents. FE, PE, and NE stand for free-exchange architecture, paid-exchange architecture, and no-exchange architecture, respectively.

In Figure 1 the average recall is plotted against the number of documents on the simulated Internet for the three different architectures. *The paid-exchange architecture shows a significantly higher recall than the no-exchange architecture, but is outperformed by the free-exchange architecture.*

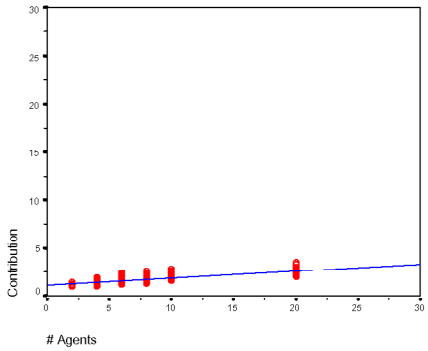
The contribution turned out to be constant for the same number of search agents, which means that the average effect an information market has on the recall of a search agent in a specific architecture is constant. We can therefore look at the development of the contribution if the number of search agents increases. Figure 2 (a and b) shows the contribution of both information markets. In both graphs, the contribution increases linearly with the number of search agents. For the free-exchange architecture the contribution increases much faster than in the paid-exchange architecture. This can be explained by the extra documents a new search agent introduces. In the free-exchange architecture, this number is equal to all the documents the search agent finds. In the paid-exchange architecture, this number equals all the documents the search agent can offer, which with the current parameter settings may result in a factor of hundred lower than in the free-exchange architecture.

The average and maximum market loads were established in an experiment with twenty search agents. As expected, the market loads in the free exchange ar chitecture were a multiple of the market loads of the paid-exchange architecture. The market loads remained approximately constant over the number of documents (on average 25 documents, with a maximum of 80 for the payed-exchange architecture, and on average 240 with a maximum of 800 for the free-exchange architecture). These observations also apply to lower numbers of agents.

Now we know that the market loads are independent of the document set size, we can investigate the relation between the market loads and the number of search agents with the expansion. This relation is important, because when the free-exchange or the paid-exchange architecture are used with a very large document

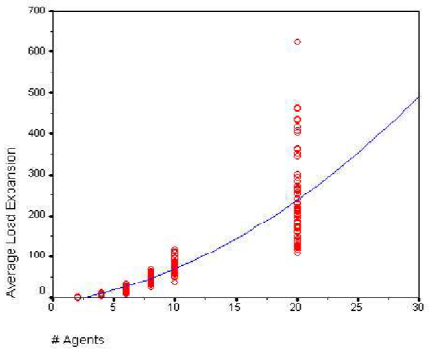


(a) The free-exchange architecture.

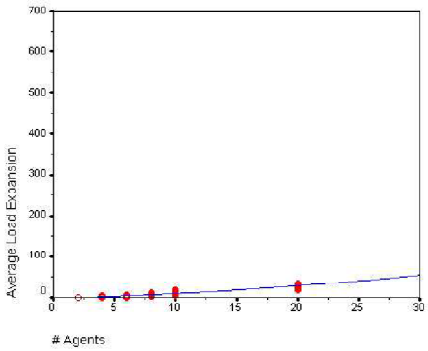


(b) The paid-exchange architecture.

Figure 2: Contribution of the information markets.



(a) The free-exchange architecture.



(b) The paid-exchange architecture.

Figure 3: The expansion of the average market load of the information markets.

set, we will need many search agents to get an acceptable recall. Therefore we want to know how the market loads develop when the number of agents increases, because the market loads indicate approximately how much memory is needed. In Section 2 we introduced the expansion, which we can now define as the market load averaged over the different document set sizes as function of the number of agents. In Figures 3 the expansions of both market loads on the information markets are shown. Examining these figures, it seems that the expansion is quadratic in the number of search agents: every search agent submits documents to every search agent in IRIS, including itself. So assume there are n search agents, the market load will be $O(n^2)$. Adding m search agents will result in a market load with $O((n+m)^2) = O(n^2 + n \times m + m^2)$. This leads to the conclusion that a linear increase in the number of search agents yields a quadratic increase in the market load. The conclusion is supported by the figures. The observations above suggest excessive memory usage for markets with large numbers of agents. However, many

documents on a market are duplicates. When represented properly, such as by references to objects, very little space is required to store duplicates of a document.

5 Conclusions

We have presented two information markets for exchanging documents between search agents. Our experiments have shown that a search agent with access to an information market is more effective than a search agent without an information market. This effect increases with the number of agents. Moreover, this effect is the best exhibited when the free-exchange architecture is used. The effect is smaller for the paid-exchange architecture, which on the other hand yields a higher efficiency. Although the expansion of the market loads increases quadratically with the number of agents, we believe, by introducing the notion of duplicates, that the approach remains scalable.

By changing the parameters of the paid-exchange market, the desired effectiveness versus efficiency can be attained. They are in between the values for the no-exchange architecture and the free-exchange architectures, which are actually extreme instances of the more general paid-exchange market.

Future research will be in two directions. First, new information markets may be investigated. New ideas can be conveniently tested with the IRIS platform. Second, scalability may be further improved by distribution of information markets: markets that specialize in particular topics may improve the scalability as well as improve the agents' performance.

References

[1] C.M. Bowman, P.B. Danzig, D.R. Hardy, U. Manbe, and M.F. Schwartz. The Harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28(1–2):119–125, 1995.

[2] A. Broder, R. Kumar, F. Maghoull, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the Web. In *Proceedings of the 9th International World Wide Web Conference*, pages 247–256, 2000.

[3] R.S. Cost, S. Kallurkar, H. Majithia, C. Nicholas, and Y. Shi. Integrating distributed information sources with CARROT II. In M. Klusch, S. Ossowski, and O. Shehory, editors, *Cooperative Informantion Systems VI*, number 2446 in Lecture Notes in Computer Science, pages 194–201. Springer Verlag, 2002.

[4] A. Gallordo-Antolin, A. Navia-Vázquez, H.Y. Molina-Bulla, A.B. Rodríguez-González, F.J. Valverde-Albacete, J. Cid-Sueiro, A.R. Figueiras-Vidal, T. Kourtis, C. Xirouhaki, and M. Koubarakis. I-Gaia: an information processing layer for the DIET platform. In *ΑΑΜΑS Proceedings*, pages 1272 1279. ACM Press, 2002.

[5] S.R. Waterhouse, D.M. Doolin, G. Kan, and Y. Faybishenko. JXTA Search: a distributed search framework for peer-to-peer networks. *IEEE Internet Computing*, 6(1):68–72, 2002.

Cooperative behavior in simulated reactive robots

D.J.M. Willems W.F.G. Haselager

Dpt. of AI / Cognitive Science, NICI - University of Nijmegen,
P.O. Box 9104, 6500 HE Nijmegen, The Netherlands

Abstract

An application was developed with which one can create and test simulations of reactive robots in simple environments. By means of this application we investigated the development of apparently plan-like collective behavior that originated out of simple behavioral structures without involving central plans or strategies. Under the appropriate circumstances seemingly cooperative and strategic behavior emerged.

1. Introduction

One of us (Willems, [8]) developed a Java/XML based application for constructing simulated robots and environments. This application allows the user to create different kinds of robots and environments and is available for use over the internet (http://enterprise2.cogsci.kun.nl/robotics/). In this paper we will examine the interaction between two kinds of simulated robots, the Unies and the Wumpi, in different environments. Unies are autonomous robots that the Wumpus can eat. A Wumpus remains near the food sources which the Unies need to sustain themselves. The Wumpus is faster than the Unies, and thus it seems that one of the few ways for the Unies to get access to the food is by means of some sort of cooperation, e.g. *decoy behavior*, in which one of the Unies entices the Wumpus away from the food, so that the others can eat (see figure 1).

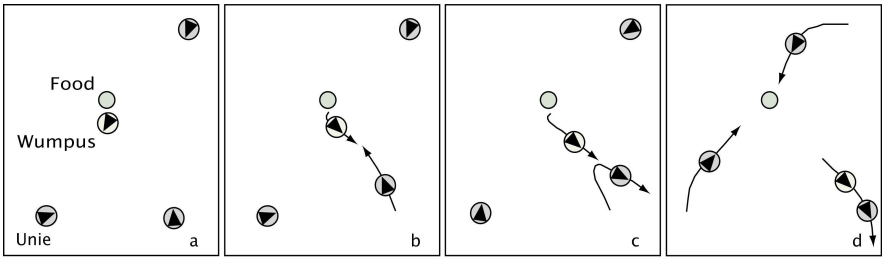


***Figure 1**: Decoy Behavior*
These diagrams show the expected decoy behavior of the Unies and a Wumpus near a Unie-food source. a) The Unies approach a food source but hold back because the Wumpus is near the food, b) The hungriest Unie comes near enough to the Wumpus for it to start the chase. c) The Unie senses that the Wumpus comes too near and reverses its direction. The Wumpus gets nearer to the Unie. d) The Wumpus chases one of the Unies. Because the Wumpus is farther away from the food, the Unies that are not being chased can approach and eat the food.

This type of behavior is interesting in that it seems to require collective plan formation and role taking. Thus, representational resources and communicative abilities seem to be needed. However, we aimed at implementing a non-representational approach to the design of the two kinds of robots using the subsumption architecture put forward by Brooks [1].

Our goal in this paper is to show that apparent cooperative behavior can be the result of the complexity of the environment. In this we attempt to follow Herbert Simon who claimed that the complexity of the behavior of the system (e.g. an ant on a beach) need not result from the complexity of the system, but rather can be a reflection of the complexity of its environment [6].

2. The simulation

In this section we will give a description of the application with which we created our simulations. A more exhaustive description is given in the user manual [7]. The application generates simulations of variable duration. It calculates the position of every object per time step. The result of this process is a simulation of the movement of objects in the environment. The simulation is rendered in two dimensions, but the simulation is actually three dimensional, objects do have height. Output of the application is in the form of a movie-like animation that can be viewed with the Play Back Panel (see figure 2).

The Play Back Panel has a complex user interface. The top part shows the environment containing obstacles, robots and food. Below the environment are the three time controlling buttons, the Start, Pause, and Stop buttons. At the top right of these buttons is a slider controlling the magnification of the environment. Below these controls is the time line that shows the camera events (zooming and moving), annotations, and movies. The application provides the possibility of annotating the simulation (providing time-dependent remarks) and of extracting movies in QuickTime format, as well as cartoon-like illustrations (such as figure 5 in this paper) that can be used in web sites, presentations and papers.

2.1. Robots and environments in XML and Java

The robots and environments are defined in XML files that can be created and modified by the user. The properties of the robots that can be defined in the definition file are for instance the size and height of the robot, their sensors and effectors, the color with which the robot is represented in the simulation, and most importantly the structure of the behavioral control system.

When a definition file is imported into the application the behavioral elements are first constructed out of the attributes that are indicated in the file, and then these elements are connected to each other as prescribed. The different behavioral elements are defined in Java. The behavior of separate behavioral elements cannot be changed within the definition file. If the user wants to add new behavioral elements such as new finite state machines, then he has to define these himself by implementing a new Java class that extends the abstract FiniteStateMachine class. For a more exhaustive explanation see the user manual of the application [7]. Environments are characterized in terms of kind and amount of obstacles, and amount of food.

Obstacles may vary in size from a bit smaller than a robot to much larger than a robot. The user can define multiple sets of obstacles with specific locations and standard deviations in the environment. The object definitions have to be written in Java and extend (subclass) the Java class Obstacle.

2.2. Simulations of reactive (subsumption architecture based) robots

For the control of the robot's behavior, we used the subsumption architecture as developed by Brooks [1, 2], the basic features of which we assume to be well known. We will limit ourselves here to a brief presentation of the control architecture of the Unies that we used in our simulations.

Four sensors were implemented for our simulations: a smell sensor to locate food and robots, a sonar sensor for avoiding obstacles, a touch sensor, and an energy sensor,

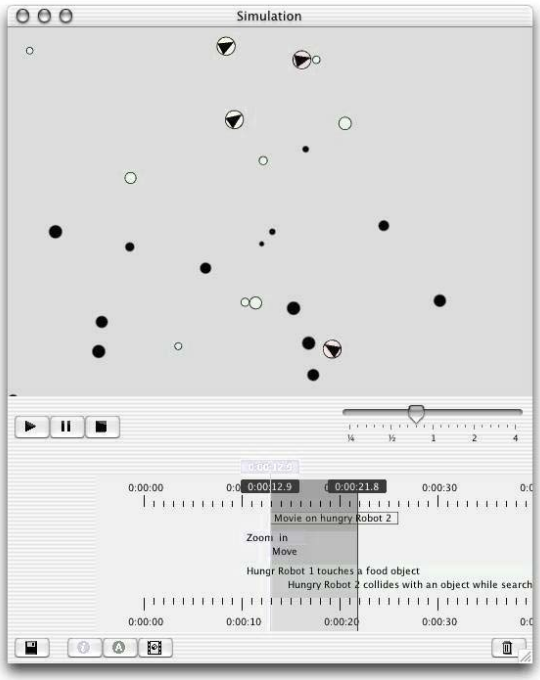


***Figure 2**: The Play Back Panel*

to measure how much energy is available to the robot. The robots have an energy depot which determines how much energy they have left. If the robot is eating food, the energy level will increase; when the energy is depleted the robot dies. The robots have three types of effectors; a speed effector for forward motion, a rotational effector for changing direction, and a feeding effector.

The Unie consists of behavioral layers that are constructed out of interconnected finite state machines. Interesting in this particular design is the finite state machine that is able to decide when the feeling of hunger takes precedence over the need to flee from the Wumpus (in the 'Wumpus' layer, see figure 3).

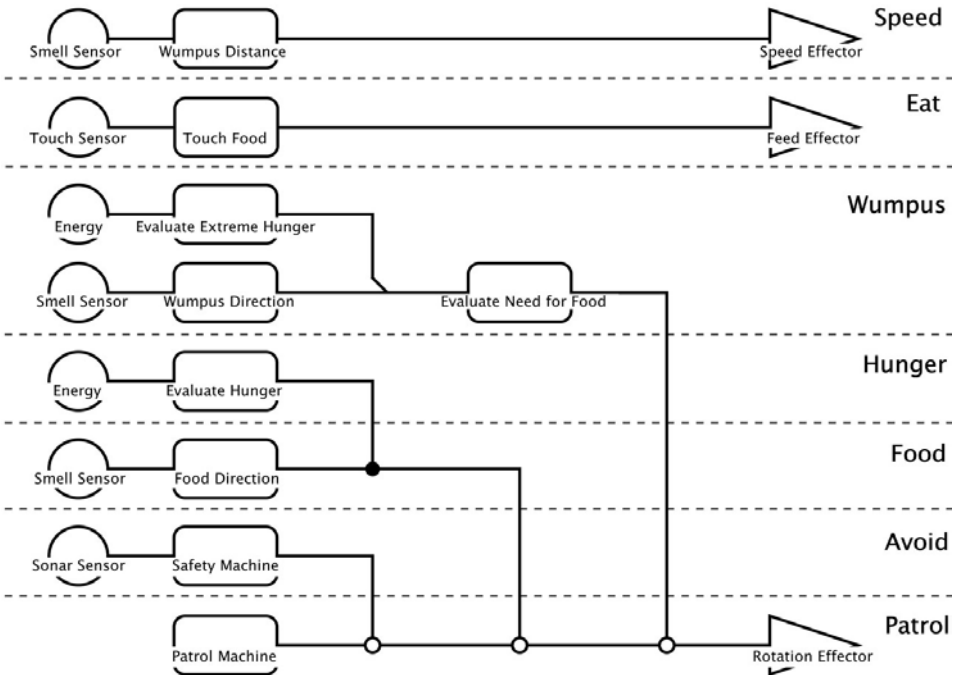


Figure 3: *The Unie architecture*

It makes this decision on the basis of the strength of the smell of the Wumpus and its relative energy level (relative to its maximum energy). The messages from the Evaluate Extreme Hunger machine are added to the messages from the Wumpus Direction machine. Both messages are then received by the Evaluate Need for Food machine that determines whether the original message from the Wumpus Direction machine gets passed on to the Patrol layer (see figure 3).

3. Types of behavior measured

To count a situation as a decoy situation, the following two conditions must be met. First, the Wumpus must be lured away from the food source. This implies that the Wumpus must be close to the food source when starting to chase a Unie. Second, at least one of the other Unies should be able to eat while the Wumpus is chasing a Unie. The decoy Unie may be caught by the Wumpus or it may escape. The first of these situations (when the Unie is caught) was described as *sacrifice* behavior and the second as *escape* behavior. Both of these situations were evaluated as an instance of *decoy* behavior, but they were counted individually. We also measured the relative occurrences of *useless death*s; the number of times a Unie lured the Wumpus away from the food source and was caught, but no Unies were able to eat. Thus, we have three situations for which we specifically searched in the test simulations (see figure 4). Moreover, we measured the times of death of each robot and the type of death of each robot (random death, hunger or as a decoy Unie). Actions such as the Wumpus chasing a Unie, or a Unie eating a food source were automatically noted by the simulation application.

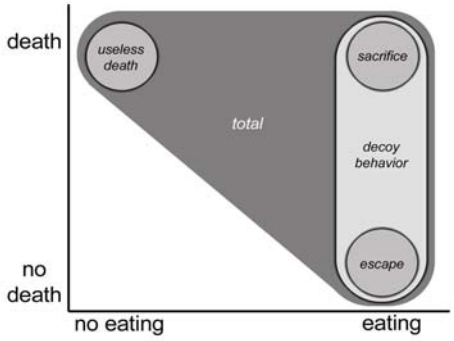


***Figure 4**: Types of behaviors*

4. The simulations

Five simulations were run per simulated environment, each with six Unies and one Wumpus. We simulated four different environments, differing in number of objects and food sources (see table 1). In the remainder of this paper we will limit our presentation to the rain-forest and desert environments.

	Number of objects		
	Food Sources	Obstacles	Surface-Type
Environment			
Desert	1	0	normal
Plains	2	0	normal
Forest	5	40	normal
Rain Forest	10	150	normal

***Table 1**: The properties of the simulated environments*

Before going on to the quantitative analyses of the results, we will present a representative case of a simulation in some detail (see figure 5).

The Unies in figure 5 show the sacrifice type of decoy behavior in a desert environment. In the first frame you can see the Wumpus circling the food source. The top left Unie is hungry and will approach the food source close enough for the Wumpus to detect the Unie (frame 2). The Wumpus will start to chase the Unie which will try to escape. It will eventually catch the Unie because the Wumpus is faster (frame 4). The Unie is caught far away from the food source, thus the other Unies can approach the food. Only one of the three Unies gets a chance to eat because it blocks the food source for the other Unies (frame 5). When the Wumpus has finished eating it will return to the food source chasing the Unies away. The Wumpus will then resume its circling behavior around the food source until one of the Unies is again hungry enough to approach the food (frame 9).

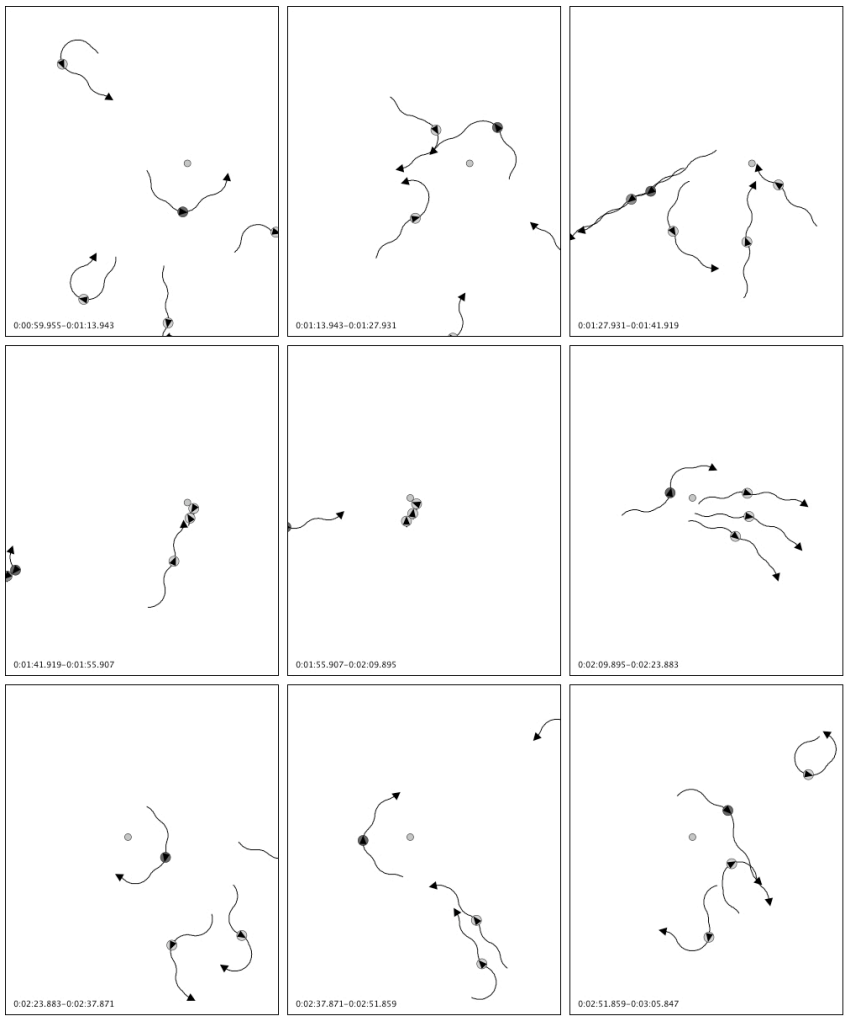


Figure 5*: Behavior of the Decoy Unie*

4.1. Results

The rain forest environment is the most complex of all tested environments because it has many obstacles and food sources. This had major consequences on the behavior of the robots. Because of the very large number of obstacles the Wumpus was not able to catch any of the Unies in any of the simulations except when it was placed near to the Unie at the start of the simulations (random deaths). The Unies were not very successful either, as they were constantly hindered by the obstacles when approaching the food sources.

In the desert environment, the number of Unies dying of hunger increases with simulation time because more time has passed for the Unies to completely deplete their energy (see table 2). Unies mainly die of hunger when they are too far away from a food source to smell it. The Wumpus always dies of hunger after having eaten all the Unies. Most Unies die after being chased away from the food. This may or may not give other Unies the opportunity to eat (which is another defining feature of decoy behavior).

Table 3 presents the absolute and relative occurrences of escape and sacrifice behavior (being the two variants of decoy behavior) in both the desert and rain forest environments. The number of times we saw escape behavior is zero. This is not completely surprising because the Wumpus is much faster than the Unies. Unless it is hindered by an obstacle, it will always catch the Unie. Although there would have been chances for a Unie to escape in the rain forest, a chase never occurred due to the richness of the food and the many obstacles preventing the Wumpus to detect a Unie near the food source it was guarding. In the desert environment we did see a number of chases, resulting sometimes in decoy behavior of the sacrifice type. Still, in many instances the Unies simply do not get enough time to eat the food. It is likely that the amount of escape and sacrifice behaviors would grow considerably once the speed of the Wumpus is decreased. At this point in time, however, this has not been simulated and tested yet.

	Time of death [s]		Type of death [%]		
	Average	Standard deviation	Random	Hunger	Chased
Unie					
First	34.0	33.8	40	0	60
Second	94.6	15.2	0	0	100
Third	201.0	73.1	0	20	80
Fourth	242.8	57.3	0	20	80
Fifth	350.5	81.2	0	40	60
Sixth	450.5	69.1	0	40	60
Wumpus	737.5	126.6	0	100	0

***Table 2**: Time and type of death in the desert environment*
Presented are the times of death (in seconds after the start of the simulation) and the type of death of each Unie and of the Wumpus. The labels first, second, etc. do not denote a specific Unie, just the sequence in which they died. One of the death types is labeled 'Chased', indicating that the Unie dies after being chased and caught by the Wumpus after it tried to approach the food.

	Occurrence			
	Desert environment		Rain forest environment	
	Absolute	Relative [%]	Absolute	Relative [%]
Escape	0	0	0	0
Sacrifice	6	25	0	0
Useless Death	18	75	0	0
Total	24	100	0	100

***Table 3**: Decoy behavior in the desert and rain forest environment*

5. Conclusion

This research project indicates that behavior that appears to be based on plan-related, central or global representations can actually be the result of the interaction of (cognitively) simple systems with each other and their environment.

The apparent cooperative behavior we observed in 6 simulations is not the result of some collective plan formation or role-division. The Unies do not use a representation of their world to guide their actions nor do they actively communicate with each other to determine the role each one has to play. They only react to the input from their senses. Yet, apparently cooperative behavior did occur. Interestingly, it seems that *scarcity* leads to cooperation. In the rain forest, Unies were not as much pressed to 'cooperate' in order to get to the food, as they were in the desert environment. In this sense, then, it is the nature of the environment that shapes the overall behavior of the simulated robots. This is in line with current ideas [3, 4] in cognitive science that emphasizes the importance of the embodied *embeddedness* of creatures for understanding their behavior.

References

[1] Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, 47, 139-159.

[2] Brooks, R. (1999). Cambrian Intelligence. Cambridge, MA: MIT Press.

[3] Chiel, H.J. & Beer, R.D. (1997). The brain has a body: adaptive behavior emerges from interactions of nervous system, body and environment. *Trends In Neurociences, 20*(12), 553- 557.

[4] Clark, A. (1997). *Being There: Putting brain, body and world together again.* Cambridge, MA: MIT- Press.

[5] Russell, S.J. & Norvig, P. (1995). Agents that Reason Logically. *Artificial Intelligence: a Modern Approach.* (151-184). Englewood Cliffs, NY. Prentice Hall, Inc.

[6] Simon, H.A. (1969/1996). *The sciences of the artificial.* Cambridge, MA, USA. Massachusetts Institute of Technology.

[7] Willems, D.J.M. (2002) *Simulation in Behavioral Robotics: The user manual.* Nijmegen, The Netherlands. http://enterprise2.cogsci.kun.nl/robotics/ UserManual/manual.html

[8] Willems, D.J.M. (2003) *Apparent Cooperative Behavior in Simulated Reactive Robots.* Nijmegen, The Netherlands. MA-Thesis, University of Nijmegen

Notes

[i] The names are mainly based on the Wumpus world, a simple environment used by Russell and Norvig [5] for demonstrating the virtues of logical reasoning in artificial agen ts.

The IntelliGate Automated Image-Based Gating Algorithm for Intracoronary Ultrasound Images

Sebastiaan de Winter[a,b] Henk Koppelaar[b] Ronald Hamers[a] Muzzafer Degertekin[a] Kengo Tanabe[a] Pedro Lemos[a] Patrick Serruys[a] Jos Roelandt[a] Nico Bruining[a]

[a] Erasmus Medical Centre, Thoraxcentre, P.O.Box 1738 3000 DR Rotterdam, The Netherlands
[b] Delft University of Technology, Fac. of Information Technology and Systems, P.O.Box 5031 2600 GA Delft, The Netherlands

Abstract

Intracoronary ultrasound (ICUS) provides high-resolution tomographic images of selected segments of coronary arteries. Series of cross-sectional images are acquired with motorized pullback imaging catheters and used for quantitative analysis in intracoronary ultrasound studies (ICUS). Due to catheter displacement in the vascular lumen during the cardiac cycle the images that are typically acquired at 0.5 mm/s are anatomically shuffled. This results in a saw-tooth shaped appearance of the coronary segment in longitudinal reconstructed views (L-views) used frequently in quantitative coronary ultrasound (QCU) software. This paper describes a novel image-based selection method (signal gating) called "Intelligate", which overcomes this problem by automatic retrospective selection of end-diastolic frames from pre-recorded ICUS studies. Our evaluation shows that there are no quantitative differences between analysis results of hardware ECG-gated and intelligated ICUS studies.

1. Introduction

Intracoronary ultrasound (ICUS) gives insight into the composition and extent of atherosclerotic plaque[1]. Previous studies have shown that ICUS may visualize atherosclerotic plaques in angiographically normal coronary arteries[2]. ICUS is used in clinical trials to evaluate the results of novel catheter-based interventional techniques as well as pharmaceutical treatments.

Cross-sectional ICUS images are acquired during a continuous speed pullback of the catheter in the coronary artery. In quantitative analysis procedures vessel wall borders are traced in a consecutive number of images. To assess vessel wall morphology and to facilitate quantitative analysis, three-dimensional (3D) reconstruction is performed using quantitative coronary ultrasound (QCU) software that visualizes segments by longitudinal cut-planes (L-views)[5]. This procedure avoids the time-consuming manual tracing of a series of individual cross-sections. However, cyclic systolic-diastolic changes of vascular dimensions and catheter motion result in saw-tooth shaped image

artifacts in the L-views that significantly hamper the quantitative analysis. (Semi-)automatic contour detection is interfered, the analysis process becomes time-consuming and the procedure may produce inaccurate results.

Most QCU analysis software packages acquire ICUS images stored on videotape at a rate of 1 frame per two seconds, randomly within the cardiac cycle, resulting in 1 mm intervals between the frames (assuming the catheter is pulled back with 0.5 mm/s), for use in area measurement and subsequent calculation of volumetric quantitative parameters. However, it has been reported that longitudinal catheter motion of more than 5 mm may occur during the cardiac cycle[3].

If a smart selection procedure is used to reduce the number of images, referred to as gating, the amount of artifacts can be reduced. Previous studies have shown that an on-line ECG-gated pullback procedure overcomes this motion problem and allows more accurate and reproducible measurements[3, 4]. However, the technology requires expensive hardware, long setup times and considerably prolongs the acquisition procedure. Most laboratories still use non-gated acquisition and most existing image databases lack ECG-gated-data. Therefore, we developed the fully automated retrospective image-based gating method "Intelligate", that can select end-diastolic ICUS frames enabling fast and accurate analysis of ICUS studies. In this paper we describe this new method and its validation.

2. The IntelliGate Method

The Intelligate method selects ICUS images recorded in the end-diastolic phase. The rationale for selecting this phase is the mutual comparability of the images as the heart is relatively motionless here and blood flow has ceased. This means that forces originating from cardiac motion and the blood flow are no longer acting on the catheter. In the end-diastolic phase an imaging catheter that is not pulled back always resumes the same position in the lumen. During other phases in the cardiac cycle, the catheter position in the coronary artery may be different, which makes it difficult to determine the absolute distances between the consecutive ICUS frames in a non-ECG-gated study. Nevertheless, currently all QCU software packages neglect the catheter motion induced by the cardiac motion.

The Intelligate method is based on identification of the near end-diastolic images from information that is present in the images, and analyzes the changes of this information over time. The method does not need a simultaneously recorded ECG- or any other physiological reference trace.

2.1. The IntelliGate Algorithm

The Intelligate algorithm can be subdivided into three main steps: Pre-processing, Frame Classification and Final Selection. The processes can be interpreted as solving a classification problem having two disjunct classes with a discriminating property of association with the end-diastolic phase or not.

Pre-processing mostly focuses on extracting information from the image series that is used to construct feature vectors describing individual frames and sequences of frames. For some features, filtering operations such as binarization based on edge

gradients are performed to enhance important image characteristics before calculation. One of the most prominent metrics is the similarity between images, in Intelligate expressed by the normalized 2D cross-correlation having the form:

$$r = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(a_{ij}-\bar{a}\right)\left(b_{ij}-\bar{b}\right)}{\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(a_{ij}-\bar{a}\right)^2\sum_{i=1}^{M}\sum_{j=1}^{N}\left(b_{ij}-\bar{b}\right)^2}} \tag{1}$$

for *M*x*N* sized images A and B having pixels a_{ij} and b_{ij} on an 8-bit greyscale and mean grey level $\bar{a}$ and $\bar{b}$ as the mean grey level. Analogous, many other image metrics are calculated and both time-domain and frequency domain properties after application of the fast Fourier transformation (FFT) to the feature signal are used to construct the feature vector. All attributes are normalized to remove biases associated with different scales using:

$$f_i^{norm} = \frac{f_i - \mu_i}{\sigma_i} \tag{2}$$

with f_i the *i*-th feature and μ_i and σ_i the sampling mean and sampling variance respectively. From all features a best classifying feature vector is selected to reduce the number of attributes and hence the dimensionality of the problem.

Frame Classification handles the non-trivial task of judging whether the given feature vector is candidate for the end-diastolic class. Since Intelligate does not have to be trained before usage as in traditional machine learning, classification cannot be done by pattern matching using comparison to known vector patterns. Instead, relational properties and the degree of agreement between feature attributes are used in a statistical framework to quantify the confidence for a possible selection of an end-diastolic frame. This decision is strongly dependent on the a priori knowledge of general motion dynamics of the human heart and its related periodic behaviour. Classification is carried out with Nearest neighbour search based on a weighted Euclidean distance metric:

$$d\left(\vec{v}_1,\vec{v}_2\right) = \sqrt{\sum_{i=1}^{k}\omega_i\left(v_{1,i}-v_{2,i}\right)^2} \tag{3}$$

for *k* features with v_1 and v_2 feature vectors and ω_i the weight factor.

The Final Selection step searches the time-domain of the candidate feature vectors in boxed intervals determined by the periodic properties of the heart cycle to select the actual end-diastolic frames from the candidate set.

2.1.1. Internal Validation Process

To validate the internal consistency of the method for each individually processed ICUS study, the mean heartbeat frequencies before (pre-Intelligate) and after gating are calculated for comparison. The rationale for this is the fact that the heart rate

of a patient as embedded in the image sequence is an independent property and may not change after application of Intelligate. The pre-Intelligate phase examines the values of different features as found in the individual ICUS images with spectrum analysis techniques. This allows for the identification of repetition frequencies of the appearance of these image features. Peaks in the power spectrum of the feature indicate large contributions from those periodic components corresponding to the repetition frequency of the original feature. The average period corresponds to the mean heart rate. The left image of figure 1 shows that the mean heart rate is approximately 73 bpm.

The Intelligate method selects the end-diastolic frames from the non-gated ICUS study and therefore the mean heart rate can be determined from the time distances between the selected frames as well (the right image in figure 1). If this post-Intelligate value resembles the pre-Intelligate heart rate, the image selection is valid in the sense that it is able to preserve the patient's heart rate. Conversely, inconsistent selection of frames inevitably leads to different frame time distances and the mean heart rate will diverge from the pre-Intelligate heart rate. Observation of the differences between both values gives an indication of the accuracy of the Intelligate process. If a difference is found, the ICUS study is not further processed and the observer is notified.

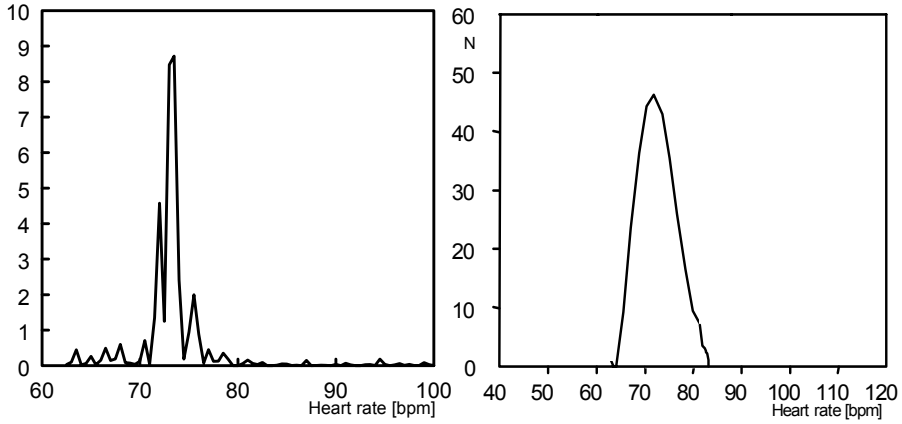


Figure 1. The left image shows a typical power spectrum derived from the ICUS study before gating. In this example, a strong peak is found at 73 bpm. On the right the distribution of the Intelligated frame rates converted to heartbeat are shown.

2.1.2. IntelliGate implementation

Intelligate was initially executed unsupervised on a dedicated SGI Octane R12000 workstation (Silicon Graphics Inc., Mountain View, CA, USA) with 1.2 GB memory running the 64-bit IRIX operating system. The method is implemented in software and operates fully automatically. Currently, the software has been ported to an array of more commonly available Intel Pentium™ IV based computer systems in a parallel multi-processor architecture (see figure 2), which can handle large volumes of ICUS studies by the parallel processing principle. Average gating speed is now less than 5,5 minutes per 1000 frames for a single processing unit.

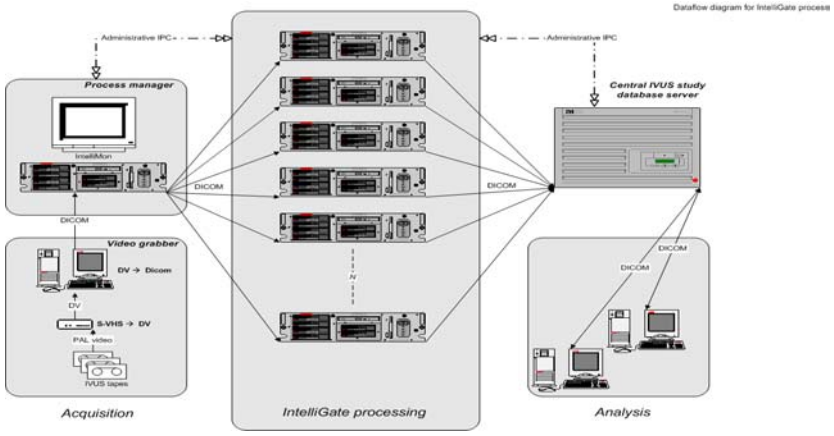


Figure 2. Multiple Processor Architecture, implemented by *n* identical Pentium™ IV based processing workstations (center). The systems are controlled by a process manager (left) that received study data from video acquisition stations. Results are stored in a central database (right).

2.2. Validation Study

2.2.1. Data acquisition

Fifteen randomly selected patients from the ITALICS trial - 6 months after implantation of a self-expanding coronary Wallstent (Schneider AB, Bülach, Switzerland) - were studied with ICUS. The ICUS studies were performed using mechanical sheath-based Boston Scientific Corporation ICUS catheters (BSC, Santa Clara, CA, USA) incorporating a 30 MHz beveled single-element transducer rotating at 1800 rpm. The sheath prevents direct contact of the imaging core with the vessel wall and increases catheter stability within the coronary artery. The ICUS studies were acquired in two passes: 1) the catheter was withdrawn at a continuous speed of 0.5 mm/s using a motorized pullback system, 2) after this, the catheter was advanced to the distal position again and an ECG-gated pullback was performed with 0.2 mm discrete intervals. The non-gated studies were stored on S-VHS videotape at a rate of 25 images/s and the ECG-gated studies digitally on Magneto-Optical disks.

2.2.2. ICUS analysis

All ICUS studies, hardware ECG-gated and non-gated after application of Intelligate (see figure 3, panel B and C), were analyzed with the CURAD (Curad BV, Wijk bij Duurstede, Netherlands) QCU analysis software[5] by 3 independent observers with comparable experience. To reduce inter-observer variability, the mean value of the quantities measured by the observers was used for comparison between the two gating techniques. The CURAD software focuses on detecting contours in L-views. In most cases 4 L-views (containing 2 contours per view) are used to complete the analysis, in case of extreme vessel eccentricity a 5th L-view (containing a 9th contour) is added. The number of L-view contours is independent of the number of included individual cross-

sections in the ICUS study. Consequently, all available cross-sectional images are used in the analysis since they all contribute to the construction of the L-view. The contours traced correspond to lumen, stent and total vessel structures. Parameters for comparison between methods were lumen, stent, total vessel volumes, stent lengths and analysis times.

2.2.3. Statistical analysis

Quantitative data are presented as mean±standard deviation. Association between continuous variables was assessed by the Pearson correlation coefficient. Comparison of continuous variables was performed by the Student's two-tailed paired t-test. A p-value < 0.05 was considered statistically significant. Relationship between two variables was analyzed by linear regression where applicable. Agreement of two methods was expressed as the mean difference±standard deviation between the two methods as proposed by Bland and Altman[6]. For the internal validation of the Intelligate method the relationship between the found pre-processing and post-processing heart rates is analyzed with linear regression and Bland and Altman analysis.

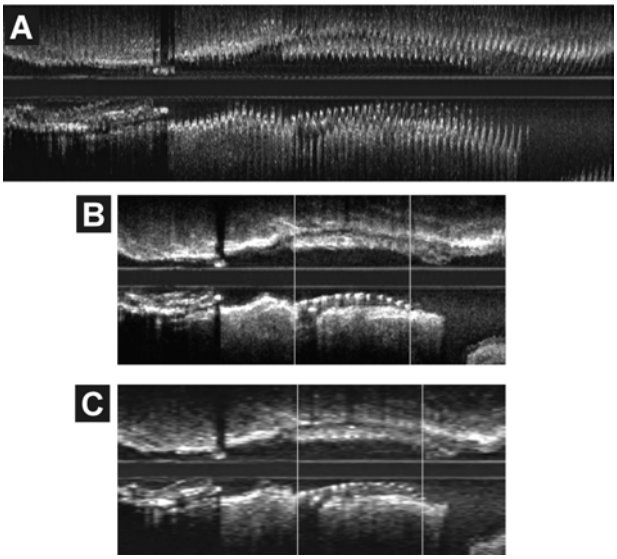


Figure 3. In panel A, a reconstructed longitudinal view (L-view) of a non-gated intracoronary ultrasound (ICUS) study is presented. Panel B shows an ICUS study of the same coronary segment but now acquired hardware ECG-gated. Finally, in panel C a L-view of the Intelligated data set is shown, that was extracted from images in the study shown in panel A.

3. Results

Stent-, lumen- and vessel volumes were not significantly different between methods (ECG-gated vs. Intelligated) and were 225±86 vs. 231±90, 159±69 vs. 160±74 and

515±248 vs. 495±245 mm^3 respectively. The difference in measured stent lengths was 28±9 vs. 27±8 mm, which was not statistically significant. There was no systematic difference for larger stents or their measured volumes. Relative differences were all within the 3,5% deviation. The Bland-Altman analysis is presented in figure 4. The required time to perform the QCU analysis was 17±4 minutes for hardware ECG-gated sets and 17±4 minutes for sets processed with Intelligate.

The internal validation of the Intelligate method for each individual processed ICUS study showed a correlation of the mean heartbeat frequencies of 0.997. Agreement between the values is expressed in a Bland-Altman analysis. Differences were calculated relative to the mean value of the two obtained values for the R-R interval time. Variability, expressed as percentage of difference between methods, was 0±1.2 %. Mean absolute difference was –4±10 ms with a standard error of the differences of 3 ms.

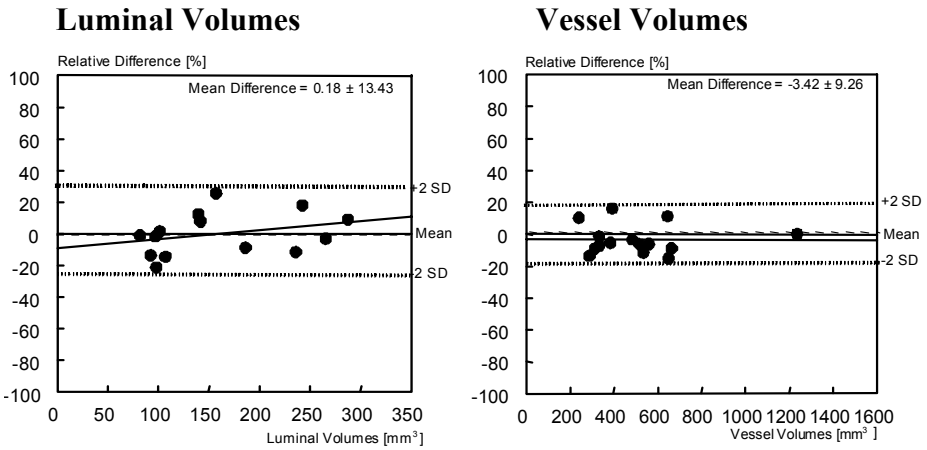


Figure 4. Bland and Altman plots for the comparison between hardware ECG-gated and Intelligated analysis results for lumen and vessel volumes.

4. Discussion

The described Intelligate method can be applied retrospectively to existing non-ECG gated ICUS studies and no additional equipment is needed in the catheterization laboratory. The current study shows that there are no statistically significant differences between the hardware ECG-gated method and the Intelligate method. One outlier in the stent length differences and one in the total vessel volume differences correspond to the same patient. In this patient, two stents have been implanted directly behind each other and the most distal stent was located in a strongly curved vessel causing difficulties in the stent length and in stent border detection.

This study also shows that QCU analysis after application of Intelligate is performed within a reasonable time as a result of the smooth appearance of the vessel wall morphology, which results in a much better performance of the applied automated contour detection algorithms. Observer interaction necessary to correct falsely detected

contours, which is usually the case in non-gated studies, is significantly reduced, saving time and reducing intra- and inter-observer variability as a consequence.

The hardware image acquisition and selection process differs in important ways from that in Intelligate. Hardware ECG-gating acquires images at 0.2 mm discrete intervals, while continuous speed pullbacks are performed at 0.5 mm/s. The average heart rate of most patients is around and slightly above approximately 60 bpm. Assuming the heart rate is 60 bpm and the pullback speed is 0.5 mm/s, Intelligate will then select end-diastolic ICUS images at 0.5 mm intervals. The longitudinal resolution of Intelligate is thus approximately half that of hardware ECG-gated studies. This did not result in any measurable difference of analysis results.

Since Intelligate can be applied retrospectively off-line, it will be possible to re-analyze ICUS studies from previous trials that have been performed with non-gated datasets.

5. Conclusion

Fully automatic retrospective image-based ICUS end-diastolic gating is feasible, resulting in similar quantitative results as hardware based ECG-gated ICUS studies.

References

[1] Fitzgerald PJ, St Goar FG, Connolly AJ, Pinto FJ, Billingham ME, Popp RL, et al. Intravascular ultrasound imaging of coronary arteries. Is three layers the norm? *Circulation* 1992;86(1):154-8.

[2] Mintz GS, Painter JA, Pichard AD, Kent KM, Staler LF, Popma JJ, et al. Atherosclerosis in angiographically "normal" coronary artery reference segments: an intravascular ultrasound study with clinical correlations. *J Am Coll Cardiol* 1995;25(7):1479-85

[3] Bruining N, von Birgelen C, de Feyter PJ, Ligthart J, Li W, Serruys PW, et al. ECG-gated versus nongated three-dimensional intracoronary ultrasound analysis: implications for volumetric measurements. *Cathet Cardiovasc Diagn* 1998;43(3):254-60.

[4] von Birgelen C, de Vrey EA, Mintz GS, Nicosia A, Bruining N, Li W, et al. ECG-gated three-dimensional intravascular ultrasound: feasibility and reproducibility of the automated analysis of coronary lumen and atherosclerotic plaque dimensions in humans. *Circulation* 1997;96(9):2944-52

[5] Hamers R, Bruining N, Knook M, Sabate M, Roelandt JRTC. A Novel Approach to Quantitative Analysis of Intra Vascular Ultrasound Images. In: *Murray A, editor. Computers in Cardiology; 2001 Sep 23-26*; Rotterdam, The Netherlands: IEEE; 2001. p. 589-592.

[6] Bland JM, Altman DG Statistical method for assessing agreement between two methods of clinical measurement. *The Lancet* 1986, i, 307-310.

Gaussian Mixture Model for Multi-sensor Tracking

Wojciech Zajdel Ben Kröse

Intelligent Autonomous Systems
University of Amsterdam, 1098SJ Amsterdam, Kruislaan 403
{wzajdel, krose}@science.uva.nl

Abstract

We present an algorithm for tracking many objects observed with distributed, non-overlapping sensors. Our method is derived from a proposition that the observations of some constant, intrinsic properties of an object form a cluster (eg. in the color space). However sensors also provide dynamic data about an object like time and location. Tracking is achieved by probabilistic clustering of observations with a Gaussian Mixture Model (GMM). We extend the GMM with auxiliary hidden variables that capture Markov dependencies between points generated by the same kernel. Since the new variables are deterministic given trajectories, the inference in our model can be done effectively with forward-backward propagation. The experiments on real and artificial data suggest that such an approach might be a faster alternative to the existing tracking methods based on MCMC sampling of trajectories.

1 Introduction

Tracking many objects with asynchronous, non-overlapping sensors is a relevant problem in automated surveillance systems. It deals with the association of data arriving at different times from various sources to find an overall assignment of observations to objects. Huang and Russell [6] provide reasons why the 'standard' data association methods [1] are not sufficient for non-overlapping sensors. These methods, like Kalman filters, exploit time and position continuity of an object between observations (eg. radar air traffic monitoring). When the sensors do not overlap the data arrive asynchronously within irregular time intervals, so a different approach is required. Moreover each sensor might introduce a specific bias (eg. due to the local illumination).

The above postulate led to the development of a method [6] specific for two cameras and an assumption that objects move only in a fixed direction between them. The association is based on probabilistic appearance models. These models specify how an object should look at the second sensor given its observation at the first sensor. Pasula *et al.* [8] showed that this approach does not scale to more than two cameras. Instead, to model probabilistic relations among observations of a single object, they saw these observations as a sequence generated by a hidden Markov model (HMM) [9]. Here the difficulty stems from the exponential number of possible sequences that can be built out of a set of all available data. For

the vehicle traffic domain (objects do not reappear at a sensor and move in one direction) Pasula *et al.* provided an efficient approximation scheme that runs in polynomial time. The approximation uses Markov chain Monte Carlo (MCMC) sampling to find a set of plausible partitions of observations into trajectories.

This paper proposes an approach for tracking by probabilistic assignment of observations into classes. We use the Gaussian mixture model as a generative model for the data and extend it to include the time- and position dependencies between observations. This idea avoids direct building of sequences, therefore saves the computational effort needed to search the space of all possible trajectories. It does not use a sampling scheme, tailor-made for specific traffic assumptions. Therefore the method is not constrained to the traffic domain.

Assumptions

There are K objects in a building that is monitored with M non-overlapping cameras. Whenever an object appears at one of the cameras we get an observation y_i of this object in the form $y_i = \{o_i, d_i\}$, where o_i describes the observed physical intrinsic properties (e.g. color or length) of an object; d_i indicates the dynamic information about an object. Typically this is the time t_i and the discrete position p_i (camera location) where the object was observed. We will assume that the dynamics d_i are observed without noise. The intrinsic properties of each object are constant. We describe them with a parameter μ_k for the kth object. The observation of intrinsic properties o_i is generated from a Gaussian distribution centered at μ_k: $p(o_i|\mu_k) = \mathcal{N}(\mu_k, R_k)$, where R_k is the variance[1]. We also assume that an object moving in a building is a 1st order Markov process. The dynamic part, like the time and position (t_i, p_i) depend only on the preceding time and position (t_{i-1}, p_{i-1}). The dependency is probabilistic, given by a distribution $p(d_i|d_{i-1})$. Another assumption is that the observations are time ordered, i.e. $t_i < t_j$ if $i < j$, for any two observations y_i, y_j.

2 Mixture model for tracking

Intuitively, one expects that the same object may appear differently each time it is observed, because of the sensor noise. However, the observations of intrinsic properties of a single object are expected to form a cluster. On the other hand, the time and position data reported by a sensor allow to see this cluster as a trajectory, adding extra dependencies (e.g. an object cannot be in two different locations p_1 and p_2 within a short time interval $t_2 - t_1$). We seek a probabilistic generative model for our observations that combines these two aspects.

Assume that for every observation y_i there is an unobserved, discrete-valued random variable x_i. The variable $x_i \in \{1, \ldots, K\}$ indicates which of the K objects generated the observation y_i, i.e., which component (kernel) $\{\mu_k, R_k\}$ generates the static part of observation y_i. To be able to generate the dynamic part of y_i,

[1]This model is rather restrictive, thus we will pre-process the observations, to remove a potential bias in the observed color arising from local illumination, see section 4.

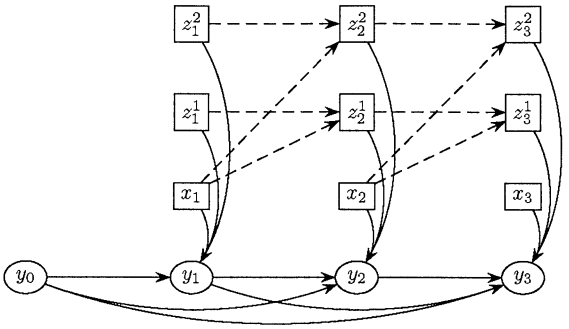


Figure 1: Dynamic Bayes Network representing the generative model with two objects. Only the first three observations are shown. Each node y_i represents an observation, node x_i indicates the assignment of the observation y_i to one of the K objects. A node z_i^k tells which of $y_{0:i-1}$ was the last observation assigned to object k before time i. This is important because each object is a Markov process.

assuming that $x_i = k$, we need to know which y was the last observation of object k (due to the assumed Markov property between observations of a single object). This could be one of the past observations $y_1, \ldots, y_{i-1}$ or nothing, denoted by an empty observation y_0. Again we pose the existence of K hidden, discrete-valued random variables $z_i^k, k = 1, \ldots, K$. A variable $z_i^k \in \{0, \ldots, i-1\}$ indicates which of the observations $y_0, \ldots, y_{i-1}$ was the last observation of object (kernel) k directly before the observation y_i.

To reason about the model we consider its graphical representation in the form of a Dynamic Bayes network of Fig.1. A solid arrow corresponds to a stochastic dependency between variables, a dashed - deterministic. As we do not know which of the past observations is assigned to the same kernel as the current observation y_i, it has to be conditioned on all of the past data[2] $y_{0:i-1}$, and the hidden pointers $x_i, z_i^{1:K}$. Let $x_i = k$ and $z_i^k = l$, the generative model for observation $y_i = \{d_i, o_i\}$ is then:

$$p(y_i|x_i, z_i^{1:K}, y_{0:i-1}) = p(y_i|x_i = k, z_i^k = l, y_{0:i-1}) = \mathcal{N}(o_i; \mu_k, R_k)p(d_i|d_l), \quad (1)$$

The value of the pointer z_i^k is deterministic given its past value z_{i-1}^k and the past x_{i-1} variable. If one knows that y_{i-1} was assigned to the object k, then this observation becomes the last observation of kth object before y_i, so the z_i^k is set to $i-1$. For other objects the last observations remain unchanged.

$$z_i^k = \begin{cases} i-1 & \text{iff } x_{i-1} = k \\ l, 0 \leq l < i-1 & \text{iff } x_{i-1} \neq k \text{ and } z_{i-1}^k = l \end{cases}. \quad (2)$$

Our model is completed with the prior distribution on indicators $p(x_i)$. It is assumed the same for every observation. We will parameterize it with K parameters: $p(x_i = k) = \alpha_k$, where $\sum_{k=1}^K \alpha_k = 1$.

[2]symbol $y_{a:b}$ denotes a series $y_a, \ldots, y_b$, similarly $z_i^{1:K}$ denotes $z_i^1, \ldots, z_i^K$.

Initially all pointers $z_1^k, k = 1, \ldots, K$ are set to zero. The probability of an observation y_i conditioned on $z_i^k = 0$, as in (1), gives the chance of starting a new trajectory with y_i. As in the standard GMM the mixture parameters $\{\mu_k, R_k, \alpha_k\}$, $k = 1, \ldots, K$ are initialized with random values, and estimated from data using EM method.

3 Approximate Inference

Given the stochastic model of Fig.1 and a set of observations $y_{1:N}$ we are interested in learning the posterior probabilities on unobserved pointers. The posterior $p(x_i|y_{1:N})$ gives the probabilistic association of observation y_i to one of the objects. It is also the necessary input for the EM [4] algorithm.

The structure of the graphical model resembles a Hidden Markov Model. The hidden terms $z_i^{1:K}$ can be predicted from the preceding $z_{i-1}^{1:K}$, x_{i-1}. However, the current observation y_i is conditioned on the current hidden variables, and all past data. Since these are observed, we can still use the standard forward-backward inference algorithm of HMM [9]. The algorithm becomes very effective by the fact that the transitions in the hidden space are deterministic. The forward pass is:

$$p(x_i, z_i^{1:K}|y_{0:i}) = \alpha p(y_i|x_i, z_i^{1:K}, y_{0:i-1}) p(x_i, z_i^{1:K}|y_{0:i-1}), \tag{3}$$

In the above formula α is a normalization term, the observation probability is given by (1), and the last term is the prediction on hidden states. We find it with:

$$p(x_i, z_i^{1:K}|y_{0:i-1}) = p(x_i) \sum_{x_{i-1}} \sum_{z_{i-1}^{1:K}} p(z_i^{1:K}|x_{i-1}, z_{i-1}^{1:K}) p(x_{i-1}, z_{i-1}^{1:K}|y_{0:i-1}). \tag{4}$$

The summation over $z_{i-1}^{1:K}$ is greatly reduced by the deterministic propagation of z variables. By fixing x_{i-1}, a configuration $z_i^{1:K}$ may be reached from only i different configurations of $z_{i-1}^{1:K}$ (see (2)).

A similar reduction occurs for the backward pass, when we update the filtered distributions $p(x_i, z_i^{1:K}|y_{0:i})$ to find the posteriors given the whole data $p(x_i, z_i^{1:K}|y_{0:N})$: (see [9, 7] for a derivation).

$$p(x_i, z_i^{1:K}|y_{0:N}) = \sum_{x_{i+1}} \sum_{z_{i+1}^{1:K}} p(x_{i+1}) p(z_{i+1}^{1:K}|x_i, z_i^{1:K}) \frac{p(x_i, z_i^{1:K}|y_{0:i})}{p(x_{i+1}, z_{i+1}^{1:K}|y_{0:i})} p(x_{i+1}, z_{i+1}^{1:K}|y_{0:N}). \tag{5}$$

The summation over $z_{i+1}^{1:K}$ involves in fact only one configuration which can be deterministically reached from the fixed x_i and $z_i^{1:K}$ (transition model of (2)).

The joint hidden state space at time i is the Cartesian product of the state space for x_i and K spaces for all z_i^k. Thus it involves $O(Ki^K)$ elements. In practice we simplified (after[3]) the representation and inference by replacing the joint posterior distribution with a factorial approximation:

$$p(x_i, z_i^{1:K}|y_{0:i}) \approx p(x_i|y_{0:i}) \prod_{k=1}^{K} p(z_i^k|y_{0:i}). \tag{6}$$

This approximation intuitively follows from the deterministic transitions in the hidden state space, and the fact that for a fixed $x_i = k$ only the kth pointer z_i^k contributes to the likelihood of observation y_i.

Once the forward and backward passes have been completed, the M step of the EM algorithm is run to update the parameters of the model. The EM update equations for our mixture components are similar to those of a regular Gaussian Mixture Model [2].

4 Experiments

We have performed a series of tests with the described model. The first compares it to the MCMC sampling[8] method, using an artificially generated data set. The second experiment demonstrates the method in the unconstrained traffic scenario, by tracking people observed with cameras in a building.

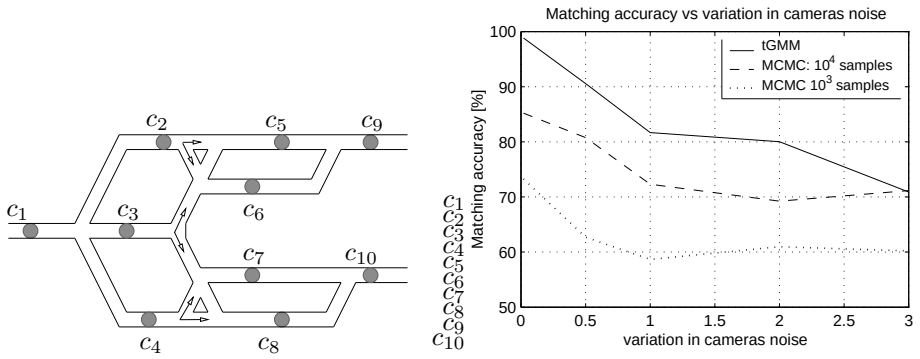
4.1 Comparison with MCMC

In this experiment we generated randomly 52 observations using a predefined network of 10 cameras as shown in Fig.2a. Every observation y_i included a simulated 1-dimensional 'color' o_i, the time of 'appearing' t_i and one of the ten positions p_i. Each camera c_n simulated a Gaussian noise added to the intrinsic color description of a simulated object. The definition of network also comprised the probabilities of transitions between camera locations $p(p_a|p_b)$, and expected travel times $\delta_{a,b}$, where a and b are camera locations. The arrival time t_i of a particular object at a camera location was distorted by noise with controlled variance $\mathcal{N}(t_i; t_{i-1} + \delta_{a,b}, R_{a,b})$. The simulated 13 objects started at camera c_1 and progressed through the network according to the arrows.

To compare the quality of the solution provided by the both methods we computed the matching accuracy of the returned trajectories. Matching accuracy of a detected track is the percentage of observations in that track that truly come from a single object to all observations in that track. The total matching accuracy of a method is the average from matching accuracies of all detected tracks.

Both methods were implemented with the assumption of a Gaussian noise. Also both methods assume Gaussian noise in the arrival time, and a known table of possible transitions $p(p_a|p_b)$. The initial setting of travel times was random, and both methods used EM to estimate the travel times $\delta_{a,b}$. In our method, we considered the kernel that had the highest posterior probability $p(x_i|y_{0:N})$ to be the final assignment of the ith observation. In the alternative method we took the solution defined by the sample configuration with the highest probability.

The results are summarized in Fig. 2b. Both methods are sensitive to a high level of camera noise. We observe that a sufficiently high number of samples is necessary for a good performance of the sampling approach. The computation time however for our method was in the range of 10^2[s], while the best MCMC run needed the order of 10^3[s]. The proposed method for low-noise data has a matching advantage, due to the fact that MCMC may get 'stuck' in a high-likelihood configuration.



a) b)

Figure 2: a) The layout of the 10 camera network, used for simulation. b) the matching accuracy of the mixture model (tGMM) and MCMC based method with different number of samples. The horizontal axis is the changing variance in the simulated noise. The experiment involved 13 objects, and a total of 52 observations.

4.2 Experiments with real data

We also test our method on real object observations, that were obtained at seven disjoint locations at the ground and first floor of the university building as in Fig. 3. In total we gathered 70 observations of 5 persons, with an equal number of observations per person.The appearance description o_i was a 9D vector containing 3D color means (RGB) computed over three regions heuristically selected[3] in the object's image. To suppress the effect of illumination on the object's observed color the original RGB representation was transformed to a normalized RGB space as proposed in [5]. The dynamic part of an observation included the time of appearance t_i, the entry and departure sides to the camera viewing fields e_i, d_i and the discrete position in the building p_i.

For this set we manually resolved the data association to have the 'ground truth' association. In the building we set a distribution $p(p_i, e_i|p_{i-1}, d_{i-1})$ giving the probability of arriving at location p_i at the side e_i when an object departs from position p_{i-1} through side d_{i-1}. In the building we also set manually the probability $p(p_1, e_1)$ of entering the building at location l_1 from side e_1. The travel times between any two locations in the building were initialized randomly and estimated together with the mixture parameters.

We observed that a variant of the EM method, called Deterministic Annealing EM [10] (DAEM) improves tracking results for this setting. Figure 4 shows a table with the result of association after learning the parameters of mixture with DAEM. In the table, the ith column presents the posterior probability on the switch variable $p(x_i|y_{0:N})$ (the darker the highest probability). Below we have included the recovered sequence of observations assigned with the first object. On

[3]The regions were three horizontal stripes of equal height going through the blob of an object.

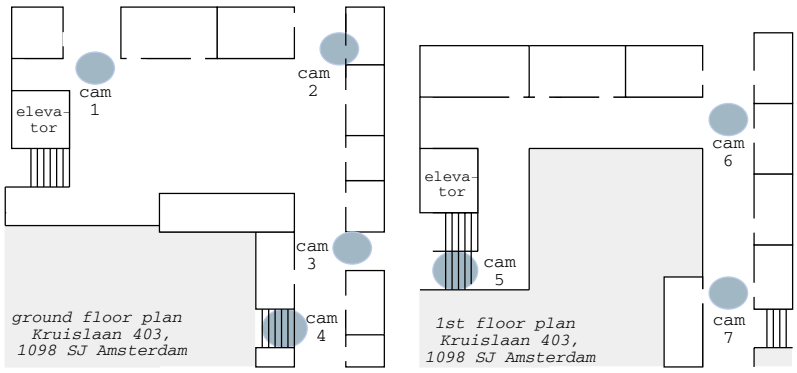


Figure 3: Building plan where the observations were taken. The gray areas show camera viewing fields.

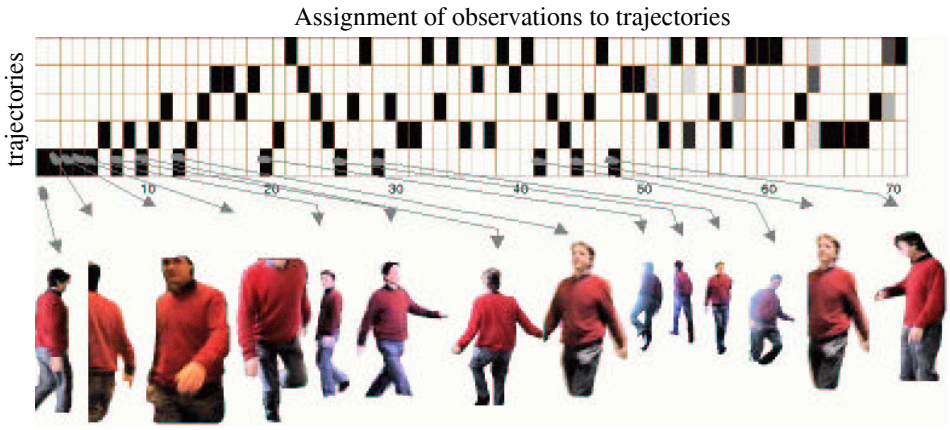


Figure 4: Posterior distribution on the hidden assignment variable x_i obtained from the set of observations. A single column is the marginal distribution $p(x_i|y_{0:N})$ shown in gray scale (dark - high probability).

average, 90% of the observations are correctly associated.

5 Conclusions

The paper proposes an alternative method for tracking multiple objects with multiple sensors. The method is based on a Gaussian mixture model for probabilistic assignment of observed data into classes, each meant to represent an individual physical object. The GMM originally developed for assignment of independent data points has been extended to handle also data connected by Markov dependencies.

The proposed idea of 'soft' tracking by a probabilistic assignment of data into tracks is in opposition to the methods that explicitly build sequences. The explicit sequences allow for easy modeling of temporal (Markovian) dependencies between data points, as for every observation its predecessor is uniquely defined. In the soft assignment there is always uncertainty about which observation is preceding the given one. For learning and inference this uncertainty has to be integrated, adding extra computational effort. Still, the experiments point out that the proposed idea may compete with algorithms approximating the space of all possible sequences.

Acknowledgment This research is supported by the Technology Foundation STW (project no ANN5312) and the Dutch Ministry of Economic Affairs.

References

[1] K. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association.* Academic Press, 1988.

[2] J. Bilmes. A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical report, University of Berkeley, ICSI-TR-97-021, 1997.

[3] Xavier Boyen and Daphne Koller. Tractable Inference for Complex Stochastic Processes. In *Proc. of Conf. on Uncertainty in Artificial Intelligence*, pages 33–42. Morgan Kaufman, 1998.

[4] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via EM algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38, 1977.

[5] M.S. Drew, J. Wei, and Z.N. Li. Illumination-invariant color object recognition via compressed chromaticity histograms of color-channel-normalized images. In *Proc. of Int. Conf. on Computer Vision*, pages 533–540, 1998.

[6] T. Huang and S. Russell. Object identification: A Bayesian analysis with application to traffic surveillance. *Artificial Intelligence*, 103(103):1–17, 1998.

[7] K. Murphy. Switching Kalman Filters. Technical report, U. of Berkeley, 1998.

[8] H. Pasula, S. Russell, M. Ostland, and Y. Ritov. Tracking many objects with many sensors. In *Proc. of Int. Joint Conf. on Artificial Intelligence*, pages 1160–1171, 1999. Stockholm.

[9] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In A. Weibel and Kay-Fu Lee, editors, *Reading in Speech Recognition*, pages 267–296. Morgan Kaufmann, 1990.

[10] N. Ueda and R. Nakano. Deterministic annealling variant of the EM algorithm. In *Advances in Neural Information Processing Systems 7*, pages 545–552. Morgan Kaufmann, 1995.

Part 2

Extended Abstracts

Intermediaries in an Electronic Trade Network

Floortje Alkemade [a] Han La Poutré [ab] Hans Amman [b]

[a] CWI, Amsterdam
[b] School of Technology Management, TU Eindhoven

We investigate whether intermediaries can make a profit in an information economy. We use evolutionary agent-based simulations to address this issue. We model a trade network game where boundedly rational consumers have to decide which links to form to sellers (profit maximizing producers or intermediaries). Our main conclusion is that intermediaries that have better knowledge about the market than the average consumer will continue to exist and make a profit if market dynamics are sufficiently complex.A full version of this paper will appear (2003) as "An Agent-Based Evolutionary Trade Network Simulation" in: Nagurney (ed.) Innovations in Financial and Economic Networks, Edward Elgar Publishers, Northampton (MA).

Electronic markets facilitate direct contact between consumers and producers, reducing the influence of intermediaries. On the other hand, intermediaries may be able to reduce the information overload consumers face. In this paper, we investigate whether intermediaries can still make a profit in an information economy. We consider a trade network game where cost-minimizing boundedly rational consumers, and profit-maximizing producers and intermediaries trade an information good. We use agent-based simulations to investigate the role of intermediaries in the trade network. More specifically, we study the influence of the network structure and information level of the agents on the level of intermediated trade. In each period of the game, consumers purchase a single unit of an information good. Consumers can buy directly from the producers or through an intermediary (at a possibly higher price). Production of information goods involves high fixed but low marginal costs. In this paper we assume that initial production costs are sunk and reproduction is very cheap, and we impose no capacity constraints on the number of goods a producer can supply. Trade can only occur if there is a connection, a link, between a consumer and a seller (producer or intermediary). Buyers (consumers and intermediaries) strategically decide which links to form by choosing a linking strategy from their associated strategy base. This strategy base is periodically updated by an evolutionary algorithm Producers strategically decide what prices to charge during a trade period. The trade network thus consist of consumers, producers, intermediaries and the links connecting them. The above model is initialized with a fixed user-specified number of consumers, producers, and intermediaries. Each trade period the economic agents take the following steps: (1) Producers choose their prices. (2) Intermediaries form links to producers

(based on their search strategy). (3) Intermediaries choose their prices. (4) Consumers form links to sellers (based on their search strategy). (5) Consumers buy one unit of the good from their preferred linked seller. (6) Consumers, intermediaries and producers calculate their utility/profits. (7) Consumer and intermediary search strategies are updated by the evolutionary algorithm. (8) Producers update their prices for the next period.

We investigate the influence of the "expertise" level of the intermediary—the number of links the intermediary initially sustains with producers—on the resulting trade network. We study the level of intermediation for different producer price schedules that lead to different market dynamics. At the beginning of the trade period, a producer has to decide which price to charge during that period. The prices set by the producers are the driving force behind the dynamics of the trade network. We investigate two basic adaptive price schedules, where producers adjust their prices on the basis of economic results from the previous period. In the first scenario all producers use a downward sloping demand curve (price decreases as demand increases). This reduces the trade network game to a coordination game for the consumers - our simulations show that consumers quickly solve this coordination game and learn to bypass the 'expensive' intermediary (consumer utility decreases as an increase in intermediated trade is observed). In the second scenario producers use a derivative follower (DF) algorithm to adjust their price, and dynamics then are more complex. The producer offering the lowest price may differ from period to period. Here, we find that an observed increase in intermediated trade does not coincide with a decrease in consumer utility (see Table 1 for results in a network with 1 intermediary, 40 consumers and 10 producers). This indicates that there exists a profitable niche for the intermediary.

Table 1: Averages over 40 generations 25 runs when producers use a DF pricing strategy.

Expertise Level Intermediary	*Fraction of Intermediated Sales*	*Average Consumer Utility*
no intermediary	0	0.89 (0.05)
0.2	0.03 (0.04)	0.89 (0.05)
0.4	0.09 (0.03)	0.89 (0.05)
0.6	0.10 (0.06)	0.89 (0.04)
0.8	0.19 (0.08)	0.89 (0.05)
1.0	0.25 (0.10)	0.89 (0.04)

The main conclusion of our ACE simulations is that intermediaries that have better knowledge of the market than the average consumer will continue to exist and make a profit if market dynamics are sufficiently complex. More specifically: (1) Intermediaries that are experts at finding the best price quotes can survive in an electronic trade network where consumers can also form direct links to producers. (2) Ultimately most consumers bypass the intermediary if direct trade is more profitable. (3) However, when producers change their prices in an adaptive way to increase their profits (derivative follower), consumers face a trade off between maintaining many costly links and getting the best price. (4) In those types of markets there is a profitable niche for the intermediary and we find that many consumers choose to trade through the intermediary.

Infant Directed Speech and Evolution of Language

Bart de Boer

AI-lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussel

Abstract

This is a long abstract of the paper that has been accepted to appear in the book containing selected papers from the Evolution of Language conference 2002 (Harvard, MA) [1]. The original presentation was called "Co-evolution of 'motherese' and acquisition of speech" and was co-authored by Patricia Kuhl.

1. Infant-Directed Speech

Infant-directed (ID) speech is the special speech that adults use when addressing infants. It should not be confused with the meaningless utterances that adults sometimes use towards pre-verbal infants. ID speech is by definition meaningful, real speech. It has wider and higher intonation, slower tempo, and has been shown to have better articulation than adult-directed (AD) speech. Also, it tends to be simpler grammatically and semantically, and tends to be more situated than adult-directed speech. Experiments have shown that infants prefer to listen to ID speech over AD speech. Finally, it turns out that adults addressing infants modify their speech automatically and without being aware of it. This has been shown to occur in many cultures.

These properties would make it likely that ID-speech aids learning of language. However, this is hard to test experimentally, as it would be impossible to deprive one group of infants from exposure to ID speech, and compare the results with a control group. Therefore, a computer model was constructed to compare the learnability of ID- and AD speech.

2. Learnability Experiment

In the first experiment the learnability of two datasets, one based on ID speech, the other on AD speech, was compared. The datasets were based on recordings of words spoken by ten mothers while (ID-condition) playing with their infants and while (AD-condition) talking to another adult. A number of target words (sheep, sock, shoe, containing the vowels /i/,

/a/ and /u/ in American pronunciation) were elicited using toys in the ID-condition and by the experimenter in the AD-condition.

Target words cut from these recordings were used as input to a computer model that extracted features (formant values) from the vowels in the words. These features were then used by a learning mechanism (expectation maximization of a mixture of Gaussians) to learn the average position of the vowels in the input. The learning system assumed that there would be three vowels in the input. It turned out that for all ten mothers ID speech was more learnable than AD speech.

3. Transfer Experiment

The second experiment investigated whether ID speech helps to preserve vowel systems when they are transferred from generation to generation in a population of agents. For this, a population of agents that can produce, perceive and learn vowel systems is created. In the population there are 20 adult and 20 infant agents. Adult agents have a repertoire of vowels, infants start out empty and have to learn a repertoire. In each generation there are 10 000 interactions, in which a random adult speaks to a random infant, using a random vowel from its repertoire. When speaking, adults reduce their articulations, as is the case in real speech. At the end of a generation, adults are removed, infants change into adults and new empty infants are inserted. The aim is to test how well the vowel system of the first generation of adults (that was pre-determined by the experimenter) is preserved over the generations.

Three conditions are compared: 1) infants use an innate compensation mechanism, 2) adults reduce less when speaking to infants (i.e. they use ID-speech) and 3) both mechanisms combined. It turns out that for simple (five-) vowel systems conditions 1) and 2) work equally well. However, when more complex (seven-vowel) systems are used only condition 3) works. Hence ID-speech is necessary for stable transfer of more complex vowel systems. This is supported by measurements of ID-speech that show mothers exaggerate more in languages with more vowels.

References

[1] B. de Boer (*to appear*) Infant directed speech and evolution of language. In M. Tallermann (ed.) *Proceedings of the 2002 evolution of language conference (working title)* Oxford:Oxford University Press.

The Balance between Proximity and Diversity in Multi–Objective Evolutionary Algorithms

Peter A.N. Bosman
Peter.Bosman@cs.uu.nl

Dirk Thierens
Dirk.Thierens@cs.uu.nl

Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

The full version of this paper has been published in the *IEEE Transactions on Evolutionary Computation*, **7**(2), pages 174–188, 2003.

1 Multi–Objective optimization

Multi–objective optimization differs from single–objective optimization in that a multiple of *objectives* should be optimized simultaneously without an expression of preference for any of the objectives. Often, these objectives are conflicting. An example is given by the situation in which the costs of some production process should be minimized while at the same time the pollution caused by the same process should be minimized. Such conflicting objectives give rise to the existence of sets of solutions, called Pareto fronts, that cannot be ordered in terms of preference when only considering their objective values. The goal is to find a diverse set of as many Pareto optimal solutions as possible instead of only a single one.

2 The balance between proximity and diversity

A variety of multi–objective evolutionary algorithms (MOEAs) have been proposed for solving multi–objective optimization problems. Especially more recent MOEAs have been shown to be efficient and superior to earlier approaches. In the development of new MOEAs, the strive is to improve performance. An important question however is whether we can expect such improvements to converge onto a specific MOEA that behaves best on a large variety of problems. The best MOEAs to date behave similarly or are individually preferable with respect to different performance indicators (functions that, given an approximation set $\mathcal{S}$, return a real value that indicates how good $\mathcal{S}$ is with respect to a certain feature). An approximation set is the set of all non–dominated solutions contained in the final population that results from running a MOEA. Unless enough evaluations are allowed such that the Pareto optimal front can be reached, we argue that the development of new MOEAs cannot converge onto a single new most efficient MOEA because the performance of MOEAs shows characteristics of multi–objective problems. The reason for this

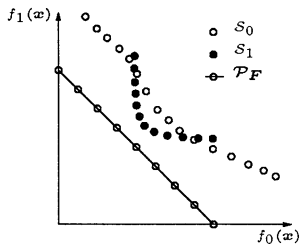


Figure 1: Whereas approximation set $\mathcal{S}_0$ is a diverse, overall good approximation of the Pareto optimal front, approximation set $\mathcal{S}_1$ is less diverse but has a better proximity with respect to the Pareto optimal front $\mathcal{P}_F$. These two approximation sets represent a trade–off in that, without a preference for diversity or proximity with respect to the Pareto optimal front, neither approximation set can be called preferable.

is that it depends on towards which goal we bias our MOEA whether we will obtain a lower proximity (closeness to the Pareto optimal front) or a higher diversity (spread of the obtained Pareto front), see Figure 1. Depending on the importance associated with diversity, a larger or smaller bias will be needed towards diversity. In this sense, there is a Pareto optimal set of MOEAs with a different emphasis on how to approach the Pareto optimal front, but such that no MOEA in this set is actually more preferable than any other MOEA in this set.

Next to a motivation of this inherent trade–off, we also point out the most important aspects for designing competent MOEAs and the impact of the trade–off on these aspects in the full version of this paper. We also present a general framework for competent MOEAs and show how current state–of–the–art MOEAs can be obtained by making choices within this framework. We further show an example of how we can separate the bias towards proximity from the bias towards diversity in the selection operator and discuss the impact of changing their ratio.

3 Experiments

We have run an existing recent MOEA [1], combined with one–point crossover, on three different multi–objective optimization problems using a fixed number of evaluations. We have varied the balance between the bias towards proximity and towards diversity using a single parameter δ that determines the ratio between selection pressure based on proximity and diversity (higher delta corresponds to more pressure towards diversity, for more details see [1,2]). The results, averaged over 10 runs, in Figure 2 clearly show a trade–off between proximity and diversity as the distance to the front worsens whereas the spread increases as δ is increased.

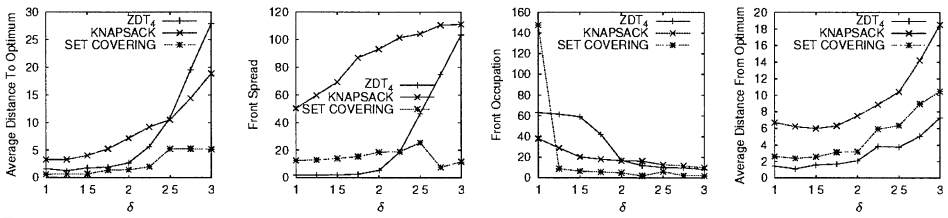


Figure 2: Results on the ZDT_4, the knapsack and the set covering problems using one–point crossover in an existing recent MOEA [1]. The results measured in four different performance indicators are shown as a function of a parameter δ that controls the balance between proximity and diversity.

4 Conclusions

The existence of the trade–off does not imply that the current state–of–the–art MOEAs cannot be improved any further. It only argues that choices between proximity and diversity will always remain. This should be kept in mind when designing new MOEAs and when comparing the experimental results of different MOEAs. For instance it is advantageous to have a MOEA in which the effort spent on diversity is separated from the effort spent on proximity. Depending on the demands of the final decision maker, such a MOEA can deal with the trade–off goals in multi–objective optimization by adjusting the ratio of these efforts.

References

[1] P. A. N. Bosman and D. Thierens, "Multi–objective optimization with diversity preserving mixture–based iterated density estimation evolutionary algorithms", *International Journal of Approximate Reasoning*, **31**, pp. 259–289, 2002.

[2] P. A. N. Bosman and D. Thierens, "The Balance between Proximity and Diversity in Multi–Objective Evolutionary Algorithms", *IEEE Trans. on Evol. Comp.*, **7**(2), pp. 174–188, 2003.

Reasoning by Assumption: Formalisation and Analysis of Human Reasoning Traces (extended abstract) [1]

Tibor Bosse[a] Catholijn M. Jonker[a] Jan Treur[a,b]

[a] Vrije Universiteit Amsterdam
Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
URL: http://www.cs.vu.nl/~{tbosse, jonker, treur}

[b] Universiteit Utrecht, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Analysis of reasoning processes has been addressed from different areas and angles, for example, Cognitive Science, Philosophy and Logic, and AI. For reasoning processes in natural contexts, which are usually not restricted to simple deduction, dynamic aspects play an important role and have to be taken into account, such as dynamic focussing by posing goals for the reasoning, or making (additional) assumptions during the reasoning, thus using a dynamic set of premises within the reasoning process. Also dynamically initiated additional observations or tests to verify assumptions may be part of a reasoning process. Decisions made during the process, for example, on which reasoning goal to pursue, or which assumptions to make, are an inherent part of such a reasoning process. Such reasoning processes or their outcomes cannot be understood, justified or explained without taking into account these dynamic aspects.

This paper presents experiments and an analysis for a pattern called 'reasoning by assumption'. This (non-deductive) practical reasoning pattern involves a number of interrelated reasoning steps, and uses in its reasoning states not only content information but also meta-information about the status of content information and about control. For this reasoning pattern human reasoning protocols have been acquired,

[1] In: R. Sun (ed.), *Proc. of the IJCAI'03 Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions*, 2003, pp. 124-129.

analysed, formalised, checked on dynamic properties and compared. To acquire empirical human reasoning traces, subjects were asked to solve a simplified game of Master Mind. The resulting traces were formalised using a temporal technique, which was already shown to be a useful analysis tool for reasoning processes in [1]. Next, these traces have been automatically analysed against dynamic properties they fulfil. To this end, a variety of dynamic properties have been specified, some of which are considered characteristic for the reasoning pattern 'reasoning by assumption', whereas some other properties can be used to discriminate between different approaches to the reasoning. For the Master Mind experiments undertaken, properties of the first, characteristic, type indeed hold for the acquired reasoning traces. Properties of the latter, discriminating type hold for some of the traces and do not hold for other traces: they define subsets of traces that collect similar reasoning approaches.

In addition to empirical traces, the analysis method can be applied to traces generated by simulation models. Dynamic properties found relevant for human traces can be used to validate a simulation model, by generating a number of simulation runs and checking the dynamic properties for the resulting traces. This type of validation has been exploited to validate a simulation model for reasoning by assumption to solve the wise men puzzle in [2].

References

[1] Jonker, C.M., and Treur, J. (2002). Analysis of the Dynamics of Reasoning Using Multiple Representations. In: W.D. Gray and C.D. Schunn (eds.), *Proceedings of the 24th Annual Conference of the Cognitive Science Society, CogSci 2002*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2002, pp. 512-517.

[2] Jonker, C.M., and Treur, J. (2003). Modelling the Dynamics of Reasoning Processes: Reasoning by Assumption. *Cognitive Systems Research Journal*, vol. 4, 2003, pp. 119-136.

Simulation and Analysis of Controlled Multi-Representational Reasoning Processes (extended abstract) [1]

Tibor Bosse[a] Catholijn M. Jonker[a] Jan Treur[a,b]

[a] Vrije Universiteit Amsterdam
Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
URL: http://www.cs.vu.nl/~{tbosse, jonker, treur}

[b] Universiteit Utrecht, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Human reasoning is often considered a process proceeding by accumulating a number of reasoning steps from start to end. An underlying assumption is that such a process can be analysed by studying each such step locally, in isolation from the rest of the reasoning process. Many reports of experimental research focus on one-trial-experiments where the number of reasoning steps is limited to one or, sometimes, at most two; e.g., [1, 3]. However, a practical reasoning process often is not a straightforward accumulation of isolated steps. First, decisions to make a reasoning step may be not a local issue at the time point of the decision, but depend on the history and goals of the reasoning process as a whole. Second, often a multitude of reasoning paths is possible; only some of these actually reach the goal. Navigation and control in the sense of making a coherent set of choices at different time points to obtain one of the successful (and preferred according to one's own characteristics) paths is a nontrivial issue. Third, during the process steps may be taken that lead to a dead end, such that the reasoning process has to reconsider these steps, leading to revision of the reasoning path. These non-local aspects of a reasoning process require specific capabilities beyond, for

[1] In: *Proc. of the Fifth International Conference on Cognitive Modelling, ICCM'03.* Universitats-Verlag Bamberg, 2003, pp. 27-32.

example, the capability to locally apply modus ponens or modus tollens. Often some form of global reasoning planning and control is performed. Decisions to make or revise a specific reasoning step are made in the context of such a reasoning plan, which also has to be taken into account as part of a reasoning state.

In many cases the same information can be represented in different manners (e.g., in arithmetic, geometric or material form). Moreover, both internal (mental) and external (e.g., written or drawn) representations may play a role. As the type of possible reasoning steps may be different for different forms of representation, these differences of representation have to be accounted for in different reasoning states. In such cases the number of possible reasoning states is not very small, and, as a consequence, the number of possible reasoning paths may be quite large. Coherent controlled navigation involving non-local aspects of decisions for reasoning steps is of major importance to deal with such a large number of possibilities.

This paper reports analysis and simulation of controlled multi-representation reasoning processes, in which the issues put forward play an important role. An analysis method for the dynamics of reasoning (adopted from [2]) is based on formal definitions of possible reasoning states and traces, and dynamic properties of these traces are specified in the Temporal Trace Language TTL. This analysis method is supported by a software environment that is able to check traces against specified dynamic properties. For simulation the component-based agent design method DESIRE is used. Traces generated by execution of a DESIRE model can be directly used as input of the analysis software environment.

References

[1] Johnson-Laird, P.N. (1983). *Mental Models*. Cambridge: Cambridge University Press.

[2] Jonker, C.M., and Treur, J. (2002). Analysis of the Dynamics of Reasoning Using Multiple Representations. In: W.D. Gray and C.D. Schunn (eds.), *Proceedings of the 24th Annual Conference of the Cognitive Science Society, CogSci 2002*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2002, pp. 512-517.

[3] Rips, L.J. (1994). *The Psychology of Proof: Deductive reasoning in human thinking*. MIT Pres, Cambridge, Mass.

Representational Content and the Reciprocal Interplay of Agent and Environment (extended abstract) [1]

Tibor Bosse[a] Catholijn M. Jonker[a] Jan Treur[a,b]

[a] Vrije Universiteit Amsterdam
Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
URL: http://www.cs.vu.nl/~{tbosse, jonker, treur}

[b] Universiteit Utrecht, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Classical approaches to representational content are based on correlations between an agent's internal state properties and external state properties. For example, the presence of a horse in the field is correlated to an internal state property that plays the role of a percept for this horse. One of the critical evaluations of this approach addresses the limitation that internal state properties are to be related to external states, and cannot be related to processes involving multiple states or events over time. Especially in cases where the agent-environment interaction takes the form of an extensive reciprocal interplay in which both the agent and the environment contribute to the process in a mutual dependency, a classical approach to representational content is insufficient. Some authors claim that it is a bad idea to aim for a notion of representation in such cases; e.g., [5, 3].

As an alternative, in [4], pp. 200-202, the approach to representational content as *relational specification* of an internal state property over time and space is advocated. Using this approach it is possible to relate an internal state property to (multiple) states at different points in time. Another perspective put forward involving multiple states over time is [1]'s *interactivist* approach. In [2] it is shown how a temporal-interactivist approach to representational content of an internal state

[1] In: R. Sun (ed.), *Proc. of the IJCAI'03 Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions*, 2003, pp. 19-28.

property can be formalised semantically based on sets of agent-environment past and future interaction trajectories or traces. Moreover, combining the temporal-interactivist perspective with the notion of relational specification, it is shown how to formalise representational content syntactically by characterizing these sets of interaction traces for a given internal state property in terms of relational specifications.

In this paper it is analysed how non-classical approaches may be used to define representational content in the case of an extensive agent-environment interplay. In particular, for a case study it is discussed how the temporal-interactivist approach and relational specification approach to representational content can be used. It is shown how to formalise the criteria for representation in terms of dynamic properties that can be formally checked for given (e.g., simulated) traces of the agent-environment interaction. Executable local dynamic properties describing basic mechanisms for the case study are presented, and simulation traces on the basis of these local dynamic properties are analysed.

References

[1] Bickhard, M.H., (1993). Representational Content in Humans and Machines. *Journal of Experimental and Theoretical Artificial Intelligence*, 5, 1993, pp. 285-333.

[2] Jonker, C.M., and Treur, J., (2002). A Temporal-Interactivist Perspective on the Dynamics of Mental States. *Cognitive Systems Research Journal,* vol. 4, 2003, pp. 137-155. Shorter version in: C. Castelfranchi and W.L. Johnson (eds.), *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'02.* ACM Press, 2002, pp. 865-872.

[3] Keijzer, F. (2002). Representation in Dynamical and Embodied Cognition. *Cognitive Systems Research Journal*, vol. 3, 2002, pp. 275-288.

[4] Kim, J. (1996). *Philosophy of Mind.* Westview Press.

[5] Sun, R. (2000). Symbol grounding: a new look at an old idea. *Philosophical Psychology*, Vol.13, No.2, 2000, pp.149-172.

Analysis of Adaptive Dynamical Systems for Eating Regulation Disorders[1]

Tibor Bosse[a] Martine F. Delfos[b] Catholijn M. Jonker[a] Jan Treur[a,c]

[a] Vrije Universiteit Amsterdam
Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
URL: http://www.cs.vu.nl/~{tbosse, jonker, treur}

[b] PICOWO, Psychological Institute for
Consultancy, Education and Research
Goeree 18, 3524 ZZ Utrecht, The Netherlands

[c] Universiteit Utrecht, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Within the context of psychotherapy often types of human behaviour and development are addressed that are highly complex, dynamic and adaptive. Recently it has been suggested that the Dynamical Systems Theory (DST, [3]) could be an adequate tool for psychotherapists to describe and analyse such behaviours; e.g., [2], [5], [6]. However, application of the DST approach in the practice of psychotherapy is not at all straightforward, and much remains to be done. A therapist's reasoning usually is performed in an informal, intuitive, partly conscious manner. Explanation of (at least parts of) this reasoning may take place in a qualitative, logical manner. In contrast, DST requires quantitative mathematical modelling, and analysis of dynamic properties is based on quantitative techniques from mathematics. This contrast between 'qualitative, logical' and 'quantitative, mathematical' makes it very difficult, if not impossible to use the DST approach as it is to adequately describe the manner in which reasoning about such an adaptive dynamical system in therapy practice takes place, or can take place in a systematic manner.

---

[1] In Proceedings of the 25th Annual Conference of the Cognitive Science Society, CogSci 2003. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2003.

Within the areas of Computer Science and Artificial Intelligence recently alternative techniques have been developed to analyse the dynamics of phenomena using logical means. Examples are dynamic and temporal logic, and event and situation calculus; e.g., [4]. These logical techniques allow to consider and relate states of a process at different points in time. This paper illustrates the usefulness of such an alternative approach for the analysis and formalisation of adaptive dynamical systems used in psychotherapy practice, in particular for the first phase of eating regulation disorders. The adaptive dynamical model put forward in [1], that describes normal functioning of eating regulation under varying metabolism levels, is formalised and used as a basis for classification of eating regulation disorders such as anorexia (nervosa), obesitas, and bulimia, and for diagnosis within a therapy. Reasoning about the dynamic properties of this model (and disturbances of them) is performed in an intuitive, conceptual semiformal manner.

References

[1] Delfos, M.F. (2002). *Lost Figure: Treatment of Anorexia, Bulimia and Obesitas (in Dutch)*. Swets and Zeitlinger Publishers, Lisse.

[2] Kupper, Z., and Hoffmann, H. (1996). A Boolean Approach to the Dynamics of Psychosis. In: W. Sulis and R. Combs (eds.), *Nonlinear Dynamics in Human Behaviour.* World Scientific, Singapore, pp. 296-315.

[3] Port and van Gelder (1995). *Mind as Motion*: *Explorations in the Dynamics of Cognition.* MIT Press, 2001.

[4] Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[5] Tschacher, W., Scheier, C., and Grawe, K., (1998). Order and Pattern Formation in Psychotherapy. *Nonlinear Dynamics, Psychology and Life Sciences*, vol. 2, pp. 195-215

[6] Warren, K., Sprott, J.C., and Hawkins, R.C., (2002). The Spirit Is Willing: Nonlinearity, Bifurcations, and Mental Control. *Nonlinear Dynamics, Psychology, and Life Sciences*, vol. **6**, pp. 55-70.

Multi-agent segmentation of IVUS images [1]

E.G.P. Bovenkamp J. Dijkstra J.G. Bosch J.H.C. Reiber

Leiden University Medical Center, Division of Image Processing
Email: E.G.P.Bovenkamp@lumc.nl

1 Methods

A novel multi-agent image interpretation system has been developed which is markedly different from previous approaches in especially its elaborate high-level knowledge-based control over low-level image segmentation algorithms. Agents dynamically adapt segmentation algorithms based on knowledge about global constraints, contextual knowledge, local image information and personal beliefs. Generally agent control allows the underlying segmentation algorithms to be simpler and to be applied to a wider range of problems with a higher reliability.

The agent knowledge model is general and modular to support easy construction and addition of agents to any image processing task. Further knowledge-based control over the image processing task has been given special consideration, since this was one of the major bottlenecks of most image interpretation systems. Agent knowledge is encoded with about 500 if-then rules, most of which are domain independent. Main knowledge modules describe how, when and where to do image processing under various circumstances, how and when to communicate, how, when and where to resolve conflicts, how to deal with multiple hypotheses and further general utilities and problem solving.

Each agent in the system is responsible for one type of high-level object and cooperates with other agents to come to a consistent overall image interpretation. Cooperation involves communicating hypotheses and resolving conflicts between the interpretations of individual agents, an example of which is given in Fig. 1.

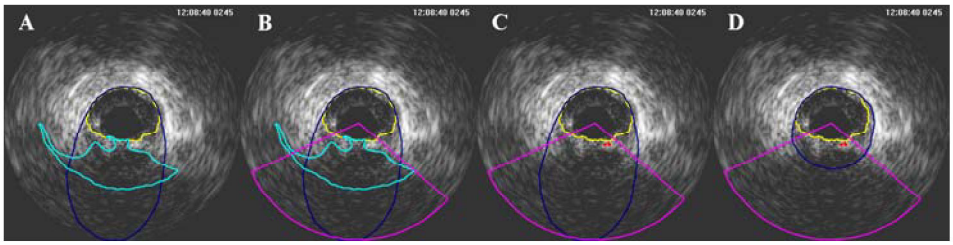


Figure 1: Multi-agent IVUS image segmentation with shadow hypothesis, sidebranch retraction and vessel adjustment. See [1] about the process.

[1] This is an abstract of [1] which will appear in a special issue of *Pattern Recognition* on Agent based Computer Vision in 2003/2004.

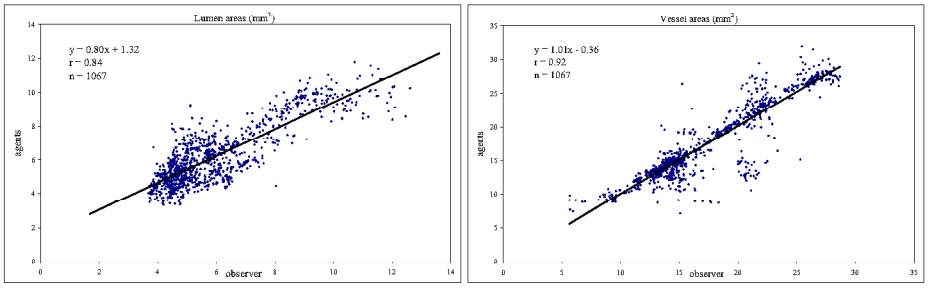


Figure 2: Comparison of the cross-sectional area (mm^2) of lumens and vessels as obtained by a trained observer and the multi-agent system.

2 Experiments and results

The system has been applied to IntraVascular UltraSound (IVUS) images which are segmented by agents, specialized in lumen, vessel, calcified-plaque, shadow and sidebranch detection. IVUS image sequences from 7 patients were processed and vessel and lumen contours were detected fully automatically. These were compared with expert-corrected semi-automatically detected contours. Results show good correlations between agents and expert with r=0.84 for the lumen and r=0.92 for the vessel cross-sectional areas, respectively.

Differences between observer and agent lumen were found at locations where sidebranches enter the main vessel. In such cases the exact lumen location is debatable. Similarly for the vessels, large shadow regions (up to 270 degrees!) make the exact location of the vessel wall difficult to guess for both agents and expert.

For the paired difference between agents and observer we found a mean difference (± 1 SD) of 0.13 $\pm$ 2.16 mm^2 for the vessel, and -0.14 $\pm$ 1.01 mm^2 for the lumen cross-sectional areas. Studies in literature which report on inter-observer variability in IVUS image segmentation show similar values. Further the Student t-test showed no statistically significant difference for the vessel areas (p=0.05). However, the difference between agents and observer was significant for the lumen cross-sectional areas ($p < 0.001$).

When compared with other studies on inter-observer variability we believe the agent system performs very well, especially when taking into account that both image acquisition and selection in those other studies were optimized in order to minimize inter- and intra-observer variability.

References

[1] E.G.P. Bovenkamp, J. Dijkstra, J.G. Bosch, and J.H.C. Reiber. Multi-agent segmentation of IVUS images. *Pattern Recognition, special issue on Agent based Computer Vision*, to appear in 2003/2004.

A Dynamic Bayesian Network for Polyphonic Music Transcription

Ali Taylan Cemgil, Bert Kappen, David Barber[†]
SNN, University of Nijmegen,
The Netherlands {cemgil,bert}@snn.kun.nl
[†]Edinburgh University dbarber@anc.ed.ac.uk

September 1, 2003

Abstract

In this paper we present a model for simultaneous tempo and polyphonic pitch tracking. The model is described in more detail in (Cemgil, et. al, *Generative Model based Polyphonic Music Transcription*, IEEE WASPAA 2003, http://www.snn.kun.nl/~cemgil). Our model, a form of Dynamical Bayesian Network [1], embodies a transparent and computationally tractable approach to this acoustic analysis problem. An advantage of our approach is that it places emphasis on modeling the sound generation procedure. It provides a clear framework in which both high level (cognitive) prior information on music structure can be coupled with low level (acoustic physical) information in a principled manner to perform the analysis. The model is readily extensible to more complex sound generation processes.

When humans listen to sound, they are able to associate acoustical signals generated by different mechanisms with individual symbolic events [2]. The study and computational modeling of this human ability forms the focus of computational auditory scene analysis (CASA) and machine listening. Recently, analysis of musical scenes is drawing increasingly more attention, primarily because of the need for content based retrieval in digital audio databases and increasing interest in interactive music performance systems.

One of the hard problems in musical scene analysis is automatic music transcription: to infer automatically a musical notation that lists the pitch levels of notes and corresponding timestamps in a given performance. However, in its most unconstrained form, i.e., when operating on an arbitrary polyphonic acoustical input, music transcription stays yet as a difficult engineering problem.

In a statistical sense, music transcription, can be viewed as a latent state estimation problem: given the audio signal, we wish to infer the underlying score (i.e. collection of onset times, note durations, pitch classes, e.t.c.). As a general inference problem, the posterior distribution is given by

$$p(\text{Score}|\text{Audio}) \quad \propto \quad p(\text{Audio}|\text{Score})p(\text{Score}) \tag{1}$$

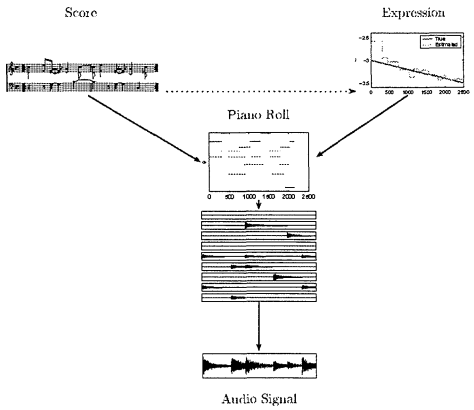


Figure 1: The generative process. The transcription is equivalent to "inverting" this generative process, given only the audio signal.

The likelihood term $p(\text{Audio}|\text{Score})$ requires us to specify a generative process that gives rise to the observed audio samples. The prior term $p(\text{Score})$ reflects our knowledge about musical scores, e.g. structure of melodies or harmony rules.

We model the likelihood term by a "two stage" generative process. The first stage is a "Score to Piano Roll" model,

$$p(\text{Piano Roll}|\text{Score}) = \sum_{\text{Expression}} p(\text{Piano Roll}, \text{Expression}|\text{Score})$$

Here, expression denotes variables that model tempo fluctuations and expressive timing deviation. The second stage is a "Piano Roll to audio", denoted as $p(\text{Audio}|\text{Piano Roll})$. This is a synthesis model, that describes how a piano roll is rendered to audio. The likelihood model is given by $p(\text{Audio}|\text{Score}) = \sum_{\text{Piano Roll}} p(\text{Audio}|\text{Piano Roll})p(\text{Piano Roll}|\text{Score})$. Conceptually, both stages are connected in a natural way through Bayes theorem. The schema of the generative process along with some preliminary results is shown in figure 1.

We describe the dynamical processes using dynamic Bayesian network [1]. We are currently investigating efficient approximation methods mainly focusing on sequential importance sampling and iterative improvement [3].

References

[1] K. P. Murphy, "Dynamic Bayesian networks: Representation, inference and learning," Ph.D. dissertation, University of California, Berkeley, 2002.

[2] A. Bregman, *Auditory Scene Analysis.* MIT Press, 1990.

[3] A. T. Cemgil and H. J. Kappen, "Monte Carlo methods for tempo tracking and rhythm quantization," *Journal of Artificial Intelligence Research*, vol. 18, pp. 45–81, 2003.

Role-assignment in open agent societies

Mehdi Dastani[a] Virginia Dignum[a,b] Frank Dignum[a]

[a] University of Utrecht, P.O.Box 80089, 3508 TB Utrecht
{mehdi, virginia, dignum}@cs.uu.nl

[b] Achmea BV, Zeist

Abstract

Original reference: M. Dastani, V. Dignum, F. Dignum: Role-assignment in Open Agent Societies. In: Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03), Melbourne, July 2003, ACM Press, 2003.

1. Extended Abstract

In the emerging model of electronic business, independently developed agents enter and leave electronic societies or marketplaces at will, in pretty much the same way that human investors enter and leave different financial markets today [1]. This creates the need to define precisely what it means that an agent "takes up" a role and "enacts" it. In this paper we present ongoing research on the determination of the conditions under which an agent can enact a role and what it means for the agent to enact a role. Open systems are characterised by heterogeneity of participants, limited trust, conflicting individual goals, and a high probability of non-conformance to specifications [2]. However promising as models for open societies, due to their ability to dynamically reorganise themselves, most existing MAS architectures are closed, in the sense that participating agents must be designed following a given internal architecture. In our opinion, the effective design of open agent societies requires the following aspects to be considered:

1. Formal frameworks are needed that specify the society structure and goals with verifiable and meaningful semantics in a way that is independent from the participating agents.
2. Specification of mechanisms through which prospective participants can evaluate the characteristics and objectives of society roles, in order to decide about participation.

3. Tools for individual agents to adapt their architecture and functionality to the requirements of an assumed role.

The importance of the first point has been widely acknowledged and extensive research is currently being done in the area of agent societies. The second point refers to the need to match agent and role objectives and functionality. At the moment most approaches to this problem simply design agents from scratch so that its behaviour complies with the behaviour described by the role(s) it will take up in the society. Comprehensive solutions for this problem require complex agents that are able to reason about their own objectives and desires and thus decide and negotiate their participation in a society. A first step on the road to this solution (taken in this paper) is to have a formalism to compare the specifications of agents and roles and determine whether an agent can enact a role. In the future, agents themselves will be able to use this mechanism to automatically evaluate their participation on a society. Finally, the third point above indicates that, once a decision has been reached that an agent will indeed enacts a role, there must be ways to modify that agent in order to include the characteristics of the assumed role. A possible solution for this point is to extend agents with an interface to the society. However, the consequence of an agent adopting a role is more drastic than this. The actual agent behaviour must be changed according to the goals, norms and reasoning rules specified by the role. Our proposal allows for different approaches to such modification of an agent, which result in different role performances. For instance, some agents will uniquely attempt to achieve the goals of its adopted role and forget its own private goals, while others will only attempt to achieve the goals from the role after its own goals have been satisfied. The paper discusses extensively the effects and possibilities for role enactment by agents.

References

[1] C. Dellarocas and M. Klein. *Civil agent societies: Tools for inventing open agent-mediated electronic marketplaces.* In Proc. AMEC (at IJCAI'99), Stockholm, Sweden, 1999.

[2] A. Artikis and J. Pitt. *A formal model of open agent societies.* In Proc. Autonomous Agents 2001, pages 192-193, 2001.

Programming Agent Deliberation

An Approach Illustrated Using the 3APL Language

Mehdi Dastani Frank de Boer Frank Dignum John-Jules Meyer

University of Utrecht, P.O.Box 80089, 3508 TB Utrecht
{mehdi, frankb, dignum, jj}@cs.uu.nl

Abstract

Published in the Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03), Melbourne, ACM Press, July 2003.

1. Extended Abstract

For many realistic applications, a cognitive agent should have both reactive and deliberative behavior [1]. The first type of behavior concerns the recognition of emergency situations in time and provides rapid responses whereas the second type of behavior concerns the planning of actions to achieve its long-term goals. In order to implement an agent's deliberative behavior, its mental attitudes such as goals and beliefs as well as the interaction between these attitudes should be implemented. Issues related to the implementation of the agents' mental attitudes can be considered as object-level concerns while issues related to the implementation of agent deliberation and the interaction between mental attitudes form meta-level concerns [2].

Agent deliberation is not limited to the planning of the tasks, but it includes various types of decisions at each moment of time [3] such as how to select a goal from a set of possible goals and whether revising a plan or executing it. These types of decisions, which constitute the agent's deliberation process and determine the types of agent behavior, are usually implemented statically in an interpreter and are not explicitly and directly programmable in an agent module. For example, the decision on goal selection is usually based on an assumed predefined ordering of the goals such that the interpreter always selects the goal that has the highest rank. Of course, it may still be possible to implement some of these decisions implicitly in the agent's mental attitudes (object level) despite

the way the interpreter is implemented. For example, the decision on goal selection can be implemented by making goals conditional on beliefs, but as we will argue, this type of approach shifts the problem rather than solves it. Moreover, we believe that doing so violates the basic programming principle called *separation of concerns*. Therefore, we aim to distinguish the object-level concerns, which are related to the mental attitudes of agents, from the meta-level concerns, which are related to the agent's deliberation process.

In this paper, we present the specification of a programming language to implement the deliberation cycle of cognitive agents. Although this programming language is designed to implement the deliberation cycle of 3APL agents [4], it can be used for other types of cognitive agents by some modifications related to how mental attitudes are represented. The 3APL programming language provides a set of programming constructs to implement agents' mental attitudes such as its beliefs, goals, basic actions, and planning rules. The aim of this paper is to extend this language with a second set of programming constructs by means of which the deliberation cycle becomes programmable. This extension should make 3APL a programming language for cognitive agents that respects the separation of concerns principle.

References

[1] G. De Giacomo, Y. Lespérance, and H. Levesque. *ConGolog, a concurrent programming language based on the situation calculus*. Artificial Intelligence, 121(1--2):109--169, 2000.

[2] F. Brazier, B.D. Keplicz, N. Jennings, and J. Treur. *Desire: Modelling multi-agent systems in a compositional formal framework*. International Journal of Cooperative Information Systems, 6:67-94, 1997.

[3] D. Kinny. *The psi calculus: An algebraic agent language*. In J.-J. C. Meyer and M. Tambe, (eds), Intelligent Agents VIII, Agent Theories Architectures and Languages, LNAI 2333, 32-50. Springer-Verlag, 2001.

[4] K.V. Hindriks, F.S.D. Boer, W.V. der Hoek, and J.-J.C. Meyer. *Agent programming in 3apl*. Autonomous Agents and Multi-Agent Systems, 2(4): 357-401, 1999.

Relational Instance Based Regression for Relational Reinforcement Learning (Extended Abstract)

Kurt Driessens Jan Ramon

Department of Computer Science, K.U.Leuven,
Celestijnenlaan 200A, B-3001 Leuven, Belgium
{kurt.driessens;jan.ramon}@cs.kuleuven.ac.be

The full paper on this topic appears in the Proceedings of the Twentieth International Conference on Machine Learning. [1]

Q-learning [6] is a model free approach to tackle reinforcement learning problems which calculates a Quality- or Q-function to represent the learned policy. The Q-function takes a state-action pair as input and outputs a real number which indicates the quality of that action in that state. The optimal action in a given state is the action with the highest Q-value.

The application possibilities of Q-learning is limited by the number of different state-action pairs that can occur. The number of these pairs grows exponentially in the number of attributes of the world and the possible actions and thus in the number of objects that exist in the world. This problem is usually solved by integrating some form of inductive regression technique into the Q-learning algorithm, which is able to generalize over state-action pairs.

One possible inductive algorithm that can be used for Q-learning is instance based regression. Instance based regression or nearest neighbour regression generalizes over seen examples by storing all or some of the seen examples and uses a similarity measure or distance between examples to make predictions about unseen examples. Instance based regression for Q-learning has been used by [5] and [4] with promising results.

Relational reinforcement learning [3] is a Q-learning approach which incorporates a first order regression learner to generalize the Q-function. This makes Q-learning feasible in structured domains by enabling the use of objects, properties of objects and relations among objects in the description of the Q-function. In this work, we investigate the use of instance based regression in relational reinforcement learning (RRL).

To apply instance based regression in the relational reinforcement learning context, a few problems have to be overcome. One of the most important problems deals with the number of examples that can be stored and used to make predictions. In the relational setting both the amount of memory to store examples and the

computation time for the similarity measure between examples will be relatively large, so the amount of examples stored should be kept relatively small.

We propose two filtering mechanisms to limit the inflow of examples into the stored database and three different measures that can be used to remove examples form the database. The filtering mechanisms are based on both the amount of knowledge we already have in the neighbourhood of the new incoming example and the error that is made in the prediction of the new example. The measures used to remove examples are based on *error contribution*, *error proximity* and *maximum variance*. Except for the maximum variance approach, which was designed specifically for Q-learning, the suggested solutions can be used for any incremental instance based regression task.

We test the behaviour of these database management approaches and their related parameters on a very simple attribute value task and compare the new instance-based RRL-system on three blocks-world tasks with a previous implementation of RRL which uses first order regression trees [2]. Empirical results clearly show that instance-based RRL outperforms regression-tree-based RRL.

References

[1] K. Driessens and J. Ramon. Relational instance based regression for relational reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning (to be published)*, 2003.

[2] K. Driessens, J. Ramon, and H. Blockeel. Speeding up relational reinforcement learning through the use of an incremental first order decision tree learner. In L. De Raedt and P. Flach, editors, *Proceedings of the 13th European Conference on Machine Learning*, volume 2167 of *Lecture Notes in Artificial Intelligence*, pages 97–108. Springer-Verlag, 2001.

[3] S. Džeroski, L. De Raedt, and H. Blockeel. Relational reinforcement learning. In *Proceedings of the 15th International Conference on Machine Learning*, pages 136–143. Morgan Kaufmann, 1998.

[4] J. Forbes and D. Andre. Representations for learning control policies. In E. de Jong and T. Oates, editors, *Proceedings of the ICML-2002 Workshop on Development of Representations*, pages 7–14. The University of New South Wales, Sydney, 2002.

[5] W. D. Smart and L. P. Kaelbling. Practical reinforcement learning in continuous spaces. In *Proceedings of the 17th International Conference on Machine Learning*, pages 903–910. Morgan Kaufmann, 2000.

[6] Christopher Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge., 1989.

A quantitative analysis of the robustness of Knowledge-Based Systems through degradation studies

Perry Groot Frank van Harmelen Annette ten Teije

Div. of Math. and CS., Faculty of Sciences, Vrije Universiteit Amsterdam

Full paper will appear in the Knowledge and Information Systems journal.

Summary: The difference between Knowledge Based Systems (KBSs) and 'conventional' software is often claimed to be the ability to deal with incomplete, incorrect, and uncertain knowledge and data. Although nowadays this distinction is not considered to be sufficient or necessary to define a KBS, it is believed that its still an essential dimension of KBS valiadation. In this paper, we argue for the need for *quantitative analysis* of the quality of KBSs. In particular, we show how *robust behavior* in the presence of incomplete system-input as well as an incomplete and incorrect knowledge base is amenable to such quantitative analysis. Our quantitative analysis is based on the idea of *degradation studies*: analyze how the quality of the output changes as a function of degrading input. We propose a set of *general definitions* which are general enough that they can be used in similar degradation experiments by others. We show the practicality of our approach by applying it to a particular *case study* thereby yielding surpising insights into the behavior of the system under study.

Approach: The approach to quantifying the robustness of a KBS is based on a *degradation study*: gradually decrease the quality of the KBS input, and measure how the KBS output quality changes as a result. We consider the approach to be the central contribution of the paper. Of course, we must be more precise about the notion of 'quality' of the KBS input and output. We expect that each task-type will come with its own measure for input quality (for KB and data) and therefore leave it open. For the output of the KBS we assume that it can be interpreted as a *set* of answers. For many typical KBS tasks, this is a realistic assumption. The output quality is then measured using the measures *recall* and *precision*. The recall is the fraction of correct answers that the system actually computes whereas the precision is the fraction of computed answers that are actually correct. These measures are well known from the literature on information retrieval. One can also consider these measures to be gradual versions of soundness and completeness. Some balance usually has to be found between those two measure as in practice increasing one usually means the other will decrease.

Experiments: For the case study we use a KBS that classifies commonly occurring vegetation in Sourthern Germany. Given a number of observables (e.g., color of flower, size of leaves, shape of leaves, etc.) the system returns one or more plants that closely match the observables. Note that it only serves to illustrate our proposal to analyze the

robustness of KBSs through degradation studies. The important aspects of this case study are the quantities we measure and how we analyze them, not the robustness results we obtain for the specific KBS we use. Underlying our experiments lies the idea that a KBS produces an output through some algorithm that accepts some input. The latter conists of data and knowledge. For the degrading input quality we look at incomplete data input (i.e., missing observations) and an incomplete and incorrect knowledge base. Furthermore, we look at the order in which data or knowledge is changed or removed. Some examples of interesting insights we found are the following:

Data input: Using the ordering of observations obtained from the test cases we obtained that the first 6 observations did not contribute to any quality increase for either the recall or precision. Interrupting the system before is has processed 6 is therefore useless. Thereafter, for about 6 to 15 observations, both measures increased rapidly and remained stable for the remaining observations. This was surprising as most cases contain 19-30 observations whereas our experiments indicate that processing more than 15 observations is not useful as it will not increase the quality of the output of the system. Using other orderings for the observations influenced our robustnes results, however, the ordering found in the test cases and the one used by the interface were surprisingly effective when compared to a random ordering.

Knowledge Base: Changing *every* rule slightly resulted in a decrease in precision of only 0.1 (on the quality interval [0,1]). Hence, the KBS seems to be robust for small errors in its knowledge base. In other experiments we looked at the robustness of the system with respect to incompleteness by removing rules from its knowledge base. When 40% of the knowledge base was removed at random there was almost no quality loss. However, when rules were removed in a biased way (figure 1) the quality of the output already started dropping after removing 15% of the rules indicating that some parts of the knowledge base had more influence on the outcome of the system.

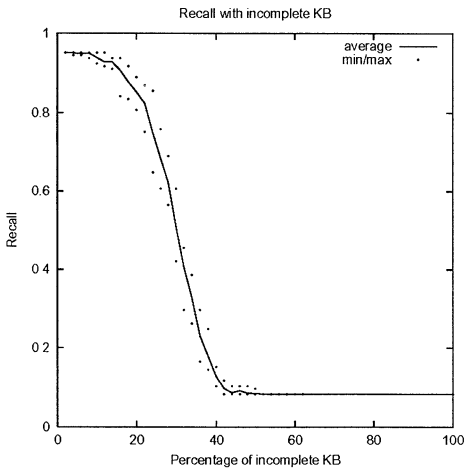


Figure 1: Recall with incomplete KB where rules were removed using a probability measure.

Final words: The ultimate suggestion that follows from this work is that any KBS should upon delivery come accompanied with a set of degradation statistics such as discussed in this paper as a quantitative way of measuring interesting and important aspects of the systems quality. this would contribute to a more empirical and quantitative analysis of AI systems and of KBSs in particular.

Updating Probabilities

Peter D. Grünwald[a] Joseph Y. Halpern[b]

[a]CWI, P.O. Box 94079, 1090 GB Amsterdam, the Netherlands
[b]Cornell University, Ithaca NY 14853, USA

This is a short overview of our paper [Grünwald and Halpern 2003].

Suppose an agent represents her uncertainty about a domain using a probability distribution. At some point, she receives some new information about the domain. How should she update her distribution in the light of this information? *Conditioning* is by far the most common method in case the information comes in the form of an event. However, there are many examples showing that naive conditioning can lead to problems. We give just one of them here.

Example 0.1: The *three-prisoners puzzle* [Mosteller 1965]: Of three prisoners a, b, and c, two are to be executed, but a does not know which. Thus, a thinks that the probability that i will be executed is $2/3$ for $i \in \{a, b, c\}$. He says to the jailer, "Since either b or c is certainly going to be executed, you will give me no information about my own chances if you give me the name of one man, either b or c, who is going to be executed." But then, no matter what the jailer says, naive conditioning leads a to believe that his chance of execution went down from $2/3$ to $1/2$. ∎

There are numerous other well-known examples where naive conditioning gives what seems to be an inappropriate answer, including the infamous *Monty-Hall puzzle* [Mosteller 1965; vos Savant 1994], the *two-children puzzle* [vos Savant 1994] and the *second-ace puzzle* [Halpern and Tuttle, 1993].[1]

Why does naive conditioning give the wrong answer in such examples? As argued by various authors, the real problem is that we are not conditioning in the right space. If we work in a larger "sophisticated" space, where we take the protocol used by the jailer (in Example 0.1) into account, conditioning does deliver the right answer. Roughly speaking, the sophisticated space consists of all the possible sequences of events that could happen (for example, what the jailer would say in each circumstance), with their probability (The notions of "naive space" and "sophisticated space" are formalized in the main paper). However, working in the sophisticated space has problems too. For one thing, it is not always clear what the relevant probabilities in the sophisticated space are. For example, what is the probability that the jailer says b if b and c are to be executed? Indeed, in some cases, it is not even clear what the elements of the larger space are. Also, working

[1]Both the Monty Hall puzzle and the two-children puzzle were discussed in *Ask Marilyn*, Marilyn vos Savant's weekly column in "Parade Magazine". Of all *Ask Marilyn* columns ever published, they reportedly [vos Savant 1994] generated respectively the most and the second-most response.

in the sophisticated space may lead to computational problems, since it can be exponentially larger than the naive space. This gives some motivation for working in the smaller, more naive space whenever possible. Example 0.1 shows that this is not always appropriate. Thus, an obvious question is when it is appropriate.

The CAR condition It turns out that this question is highly relevant in the statistical areas of *selectively reported data*, *missing data* and *survival analysis*. Building on previous approaches, Heitjan and Rubin [1991] presented a necessary and sufficient condition for when conditioning in the "naive space" is appropriate. Nowadays this so-called *CAR (Coarsening at Random)* condition is an established tool in survival analysis. (See [Gill, van der Laan and Robins 1997] for an overview.) We examine this criterion in our own, rather different context, and show that it applies rather rarely. Specifically, we show that there are realistic settings where the sample space is structured in such a way that it is impossible to satisfy CAR, and we provide a criterion to help determine whether or not this is the case. We also give a randomized algorithm that generates all and only distributions for which CAR holds, thereby solving an open problem posed by [Gill, van der Laan and Robins 1997].

Jeffrey conditioning, relative entropy updating We then show that the situation is worse if the information does not come in the form of an event. For that case, several generalizations of conditioning have been proposed. Perhaps the best known are *Jeffrey conditioning* and *Minimum Relative Entropy (MRE) Updating* (also known as *cross-entropy*). We show that, just like 'naive conditioning', these update rules can be compared to ordinary probabilistic conditioning in a 'sophisticated' space. In this space the available information can be represented as an event after all, and probabilistic conditioning leads to correct (in a certain sense) inferences. We show that Jeffrey conditioning, when applicable, can be justified under an appropriate generalization of the CAR condition. Although it has been argued, using mostly axiomatic characterizations, that MRE updating (and also Jeffrey conditioning) is, when applicable, the *only* reasonable way to update probability, it is well known that there are situations where applying MRE leads to paradoxical, highly counterintuitive results, exemplified by puzzles such as the *Judy Benjamin problem* [Van Fraassen 1981, Uffink 1995, Uffink 1996]. Seidenfeld [1986] , strengthening results of Friedman and Shimony [1971], showed that there is *no* sophisticated space in which conditioning will give the same answer as MRE in this case.We strengthen these results by showing that, even in a class of much simpler situations (where Jeffrey conditioning cannot be applied), using MRE in the naive space corresponds to conditioning in the sophisticated space in essentially only trivial cases. These results taken together show that *generally speaking, working with the naive space, while an attractive approach, is likely to give highly misleading answers.* That is the main message of our paper.

References

Grünwald, P. and J. Halpern (2003). Updating probabilities. Accepted for publication in *Journal of Artificial Intelligence Research*. To appear.

A Decommitment Strategy in a Competitive Multi-Agent Transportation Setting *

P.J. 't Hoen[†] and J.A. La Poutré[†‡]

[†] Center for Mathematics and Computer Science (CWI)
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands,
phone +31 20 5929333, fax +31 20 5924199.

[‡]TU Eindhoven,
De Lismortel 2, 5600 MB Eindhoven, The Netherlands.
{hoen,hlp}@cwi.nl

Extended Abstract

Decommitment [1, 2] is the action of foregoing of a contract for another (superior) offer. It has been shown that, using decommitment, agents can reach higher utility levels in case of negotiations with uncertainty about future prospects. In this paper, we study the decommitment concept for the novel setting of a large-scale logistics setting with multiple, competing companies. Orders for transportation of loads are acquired by agents of the (competing) companies by bidding in online auctions. Using computational experiments, we find significant increases in profit that scale with the size of operations and uncertainty of future prospects when computerized agents can decommit and postpone the transportation of a load to a more suitable time . The observed profit margins in the experiments are significant from the perspective of the transportation sector where a 4% profit is considered exceptional. For example, the average profit margin before taxes for the Dutch road transport sector (from 1989 to 1999) was only 1.6%.

We observed in the computational experiments that decommitment of a load occurs predominantly when trucks are close to filling their maximum capacity. When few loads are available, the loads are almost all picked up and transported. If the availability increases, the (positive) effect of decommitment then increases, until the trucks reach their capacity limits. In the case of an excess of cargo, the added value of decommitment decreases as the tasks which are needed to fully utilize the remaining capacity of an agent are often available. Hence, we hypothesize a decommitment strategy is most beneficial when a truck is close to reaching its maximum capacity and has a limited number of extra tasks to choose from. We believe this is a general result for an agent capable of doing

*This paper is originally published at AAMAS-03 Workshop on Agent-Mediated Electronic Commerce V (AMEC-V), 2003 Melbourne, Australia

multiple tasks in parallel. This hypothesis must be kept in mind when evaluating whether to apply a decommitment strategy.

Figure 1 shows the profits made by a company (with and without the use of a decommitment strategy) as a function of the number of depots on the grid where loads stochastically appear in the course of a day of operations. The

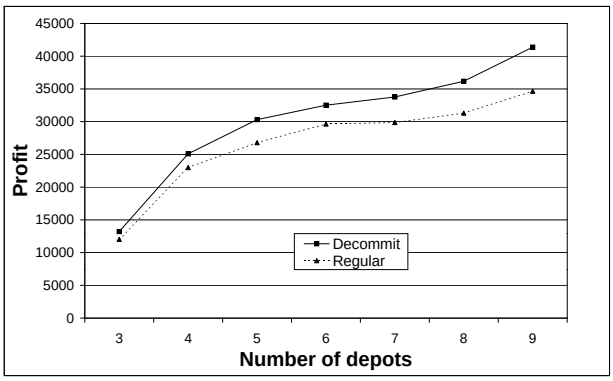


Figure 1: Profits made by a company (with and without decommitment) as a function of the number of depots on the grid.

positive effect of decommitment on a company's profit increases as the grid becomes more densely filled and the world becomes more difficult for agents to accurately predict. Results for more complex scenarios (more competing companies, more trucks per depot, prices more dependent on weight of the load, etc . . .) show similar trends such that impact of a decommitment strategy increases in importance with the complexity of the world.

It is also important to note that the use of decommitment by one company can decrease the performance of the non-decommitting companies. This loss can amount up to half the increase in profit of the company who uses a decommitment strategy. This effect is of importance when the margin for survival is small and under-performing companies may be removed from the field.

For specific applications beyond that of our model and for novel areas, the added value of decommitment, and the circumstances where it can be applied successfully should be studied further. However, based upon our computational experiments, we hypothesize that the positive impact of a decommitment strategy increases with the complexity of the operating domain, as it then becomes of greater importance to have the opportunity to roll-back a previous sub optimal decision [2].

References

[1] M. Andersson and T. Sandholm. Leveled commitment contracts with myopic and strategic agents. *Journal of Economic Dynamics and Control*, 25:615–640, 2001.

[2] T. Sandholm and V. Lesser. Leveled-commitment contracting, a backtracking instrument for multiagent systems. *AI Magazine*, Fall 2002:89–100, 2002.

Emerging shared action categories in robotic agents through imitation

Bart Jansen Bart De Vylder Bart de Boer
Tony Belpaeme

Vrije Universiteit Brussel
Artificial Intelligence Laboratory
Pleinlaan 2, 1050 Brussel, Belgium

Original paper: Jansen, B., De Vylder, B., de Boer, B. and Belpaeme, T. (2003) Emerging shared action categories in robotic agents through imitation. Proceedings of the 2nd International Symposium on Imitation in Animals and Artifacts, Aberystwyth, UK.

1 Introduction

Most of the work that has been published on imitation in robots focuses on the learning of action categories in a teacher–student context [1, 2, 4]. In such a set-up one agent acting as teacher already has a full action repertoire. And by observing the teacher executing actions, this repertoire is passed on to the student. However, such a set-up does not explain how repertoire of actions emerges and does not shed light on the dynamics of imitation.

We demonstrate a set-up in which new action categories emerge through a continuous interplay between exploring actions and imitating actions. For this we use a population of agents engaging in imitative interactions, called *imitation games*. A new action category only spreads into the population if it can be successfully imitated by the agents. However, if an action is hard to observe or to imitate, it will not be picked up by other agents. Our concept of imitation games strongly resembles the concept of imitation games used in [3] in the context of vowel systems.

A robotic agent has a stereo camera head and a robot arm. With the arm it can make different kinds of gestures which can be observed through the vision system. Gestures are restricted to motion trajectories from one point to another and do not involve manipulations of other objects and do not carry any meaning, yet.

The agents initially start with empty repertoires and gradually develop a set of action categories. Action categories contain both the motor commands required to perform the action and an observation. The storage of this observation in the category, enables the agents to recognize categories. For every game two agents are randomly selected from the population. One agent will take the role of *initiator*, the other agent will be the *imitator*. Both agents use the vision system and the manipulator. The game starts when the initiator selects a random action category and executes the action that is associated with this category. The imitator observes this action and tries to categorize the observation. The imitator imitates the initiator's action by executing the action associated with its category. The initiator observes this action and categorizes the observation. If the initial category and the category of the imitated action are the same, then the initiator decides that the game succeeds, otherwise it fails. The initiator sends non-verbal feedback about

the outcome of the game to the imitator. Depending on this feedback, the imitator adapts its categories. The result of this single imitation game is that the categories of both agents become more similar. As all agents engage many times in imitation games—both in the role of initiator and imitator—the categories of all agents will become more similar. This mechanism, combined with the occasional addition of new random action categories and the removal of previously unsuccessful categories, enables the agents to build up a repertoire of shared action categories.

2 Results

We introduce three measures to evaluate the quality of the learnt categories and present results using these measures. Preliminary simulation results show that using imitation games, shared repertoires of action categories can be obtained. These repertoires are non-trivial, as they consist of multiple action categories. They are also very successful: for populations of only two agents imitation success is always in the 90% range even though noise is present. In larger populations the imitation success is lower, but still better than random repertoires would achieve. Imitation games in this setup are based on the imitator updating its action categories. In this case calibration between the visual system and the robot arm, inverse kinematics of the manipulator and feedback between the agents about the outcome of the game are required.

In a variation to this game, we show that if the population is restricted to only two agents, action categories can emerge and be shared without calibration, without non-verbal feedback and without the agents have built-in notion of the inverse kinematics of their manipulator.

References

[1] Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn. Do as I do: Correspondences across different robotic embodiments. In D. Polani, J. Kim, and T. Martinetz, editors, *Proceedings of the Fifth German Workshop on Artificial Life (GWAL5)*, pages 143–152, 2002. 18-20 March 2002, Lübeck, Germany.

[2] Aude Billard and Gillian Hayes. Transmitting communication skills through imitation in autonomous robots. In Andreas Birk and Yiannis Demiris, editors, *Proceedings of EWLR97, Sixth European Workshop on Learning Robots, Lecture Notes on Artificial Intelligence*, volume 1545, pages 79–95. Springer, 1997. Brighton, UK, July 1997.

[3] Bart de Boer. Self organization in vowel systems. *Journal of Phonetics*, 28:441–465, 2000.

[4] P. Vogt. Grounding language about actions: Mobile robots playing follow me games. In Meyer, Bertholz, Floreano, Roitblat, and Wilson, editors, *SAB2000 Proceedings Supplement Book*. International Society for Adaptive Behavior, 2000.

Finding non-local dependencies: beyond pattern matching

Valentin Jijkoun

Language and Inference Technology Group,
ILLC, University of Amsterdam

Abstract

This paper is a summary of: Valentin Jijkoun. Finding non-local dependencies: beyond pattern matching. In *Proceedings of the ACL-2003 Student Research Workshop, pages 37-43.*

1 Introduction

Natural language parsing aims at determining syntactic structure of free text. Although much current research in the field focuses on extracting local syntactic relations from sentences, non-local dependencies have recently started to attract more attention, due to the fact that often they represent relations important for NLP applications. Non-local dependencies (also called long-distance, long-range or unbounded) appear in many frequent linguistic phenomena, such as passive, WH-movement, control and raising etc. For example, in sentence *John tried to hit the ball* the subject–verb relation between *John* and *hit* is a non-local relation.

Several approaches to handling non-local relations have been investigated in the literature. In (Clark et al., 2002) long-range dependencies are included in parser's probabilistic model, while Johnson (2002) presents a method for recovering non-local dependencies *after* parsing has been performed.

More specifically, Johnson (2002) describes a pattern-matching algorithm for inserting empty nodes and identifying their antecedents in phrase structure trees or, to put it differently, for recovering non-local dependencies. From a training corpus with annotated empty nodes Johnson's algorithm first extracts those local fragments of phrase trees which connect empty nodes with their antecedents, thus "licensing" corresponding non-local dependencies. Next, the extracted tree fragments are used as patterns to match against previously unseen phrase structure trees, which allows to identify non-local relations.

We developed a similar approach using dependency structures rather than phrase structure trees and extending bare pattern matching with machine learning techniques.

The evaluation of our algorithm on data automatically derived from the Penn Treebank shows an increase in both precision and recall in recovery of non-local dependencies by approximately 10% over the results reported in (Johnson, 2002).

2 Our method and evaluation results

Our training and test corpora of dependency structures were automatically derived from the Penn Treebank II corpus. Local syntactic structures in the Penn Treebank were mapped to local dependencies and traces (i.e. empty phrase structure nodes, possibly co-indexed with other nodes in the same structure) were mapped to non-local dependencies.

For every non-local relation in the dependency corpus we extracted its local syntactic context (i.e. the fragment of the local dependency tree, connecting the members of the non-local relation) and a number of other features (part-of-speech tags of the involved words, presence of subject/object dependents, etc.)

The information extracted from the training sub-corpus was used to train a statistical classifier. The task of the classifier was, given information extracted from a local dependency tree, to decide whether non-local dependencies should be introduced and, if so, what their labels should be.

The performance of the classifier was then evaluated on the test sub-corpus, using conventional metrics: labelled precision (the fraction of the correct dependencies among all the dependencies found), labelled recall (the fraction of the correctly found dependencies among all the dependencies) and f-score (harmonic mean of precision and recall). Compared to the results reported in (Johnson, 2002), the f-score increased from 0.75 to 0.86.

Using dependency structures instead of phrase structure trees, we also significantly reduced the number of the patterns (i.e. the fragments of local dependency trees) from around 11,000 (as reported by Johnson, 2002) to 16, which actually allowed us to use statistical classifier, since we had enough training data for every pattern. Moreover, we were able to compare the performance of the algorithm for different patterns, i.e. for different syntactic constructions, and identify syntactic phenomena difficult for our approach, that is, those where structural syntactic information alone is not enough to robustly recover non-local dependencies, and subcategorization or lexical information is required.

References

[Johnson, 2002] Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Meeting of the ACL.*

[Clark et al., 2002] Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL*, pages 327-334.

Learning the Ideal Evaluation Function

Edwin D. de Jong [a] Jordan B. Pollack [b]

[a] Universiteit Utrecht, ICS, DSS Group
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands.
[b] DEMO Lab, Volen National Center for Complex Systems,
Brandeis University MS018
415 South street, Waltham MA 02454-9110, USA

This abstract summarizes the following paper: De Jong, E.D. and J.B. Pollack (2003). Learning the Ideal Evaluation Function. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2003, pp. 277-288. Springer-Verlag, LNCS series. Available from: `http://www.cs.uu.nl/~dejong/`

Designing an adequate fitness function requires substantial domain knowledge and can be a critical factor in evolution, see e.g. [6]. Often though, tests revealing information about the qualities of individuals can readily be performed. In chess for example, absolute evaluation of strategies is extremely difficult, while comparing individuals only requires knowledge of the rules of the game. If individuals can be evaluated based on tests, coevolution can be used to circumvent the problem of defining a fitness function.

Coevolution has already produced a number of promising results [7, 10, 9]. However, there are various ways in which evaluation in coevolution can become inaccurate [12]. As a step towards accurate evaluation, Juillé defines a domain-specific *ideal trainer* [8]. Rosin provides an *automatic* mechanism for accurate evaluation, but the approach is based on a single-objective perspective, and likely to stall for problems with multiple underlying objectives. Pareto-coevolution [4, 11] uses the outcomes of a learner against coevolving evaluators (tests) as objectives in the sense of Evolutionary Multi-Objective Optimization.

By combining Rosin's complete set of tests with Ficici's important notion of *distinctions* [5], we arrive at the concept of a *Complete Evaluation Set*. The complete evaluation set was first described in [2], and detects all differences between learners relevant to selection. A related set based on order theory is described in [1].

We prove that given a complete evaluation set as evaluators, Pareto-coevolution leads to *ideal evaluation*, i.e. evaluation according to all underlying objectives of a problem. The complete evaluation set provides a practical way for coevolution methods to approximate ideal evaluation. An algorithm based on this principle is described, and found to achieve stable progress on a number of test problems that could not be addressed by standard coevolution methods used for comparison. This paper summarizes the results described in our technical report [2]. A more extensive account of this work is to appear in [3].

References

[1] Anthony Bucci and Jordan B. Pollack. Order-theoretic analysis of coevolution problems: Coevolutionary statics. In *Proceedings of the GECCO-02 Workshop on Coevolution: Understanding Coevolution*, 2002.

[2] Edwin D. De Jong and Jordan B. Pollack. Principled Evaluation in Coevolution. Technical Report CS-02-225, Brandeis University, May 31, 2002.

[3] Edwin D. De Jong and Jordan B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, (To appear).

[4] Sevan G. Ficici and Jordan B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Julian Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.

[5] Sevan G. Ficici and Jordan B. Pollack. Pareto optimality in coevolutionary learning. In Jozef Kelemen, editor, *Sixth European Conference on Artificial Life*, Berlin, 2001. Springer.

[6] Pablo Funes. *Evolution of Complexity in Real-World Domains*. PhD thesis, Brandeis University, Waltham, MA, 2001.

[7] D. W. Hillis. Co-evolving parasites improve simulated evolution in an optimization procedure. *Physica D*, 42:228–234, 1990.

[8] Hugues Juillé. *Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm*. PhD thesis, Brandeis University, 1999.

[9] Jordan B. Pollack and Alan D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(1):225–240, 1998.

[10] Karl Sims. Evolving 3D morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 28–39, Cambridge, MA, 1994. The MIT Press.

[11] Richard A. Watson and Jordan B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Julian Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.

[12] Richard A. Watson and Jordan B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-01*, pages 702–709, San Francisco, CA, 2001. Morgan Kaufmann.

Intelligent Support for Solving Classification Differences in Statistical Information Integration

C.M. Jonker [a] D. Verwaart [b]

[a] Vrije Universiteit, de Boelelaan 1081a, 1081 HV Amsterdam

[b] LEI, Burg. Patijnlaan 19, 2585 BE den Haag

Abstract

Integration of heterogeneous statistics is essential for political decision making on all levels. Like in intelligent information integration in general, the problem is to combine information from different autonomous sources, using different ontologies. However, in statistical information integration specific problems arise. This paper is focused on the problem of differences in classification between sources and goal statistics. Comparison with existing information integration techniques leads to the conclusion that existing techniques can only be used if individual data underlying the statistics is accessible. This requirement is usually not met, due to protection of privacy and commercial interests. In this paper a formal approach and software tools are presented to support statistical information integration, based on a generic ontology for descriptive statistics, and heuristics that work independent of the domain of application. The heuristics were acquired from economic experts working in the field of European Common Fisheries Policy.

1. Introduction

Integration of heterogeneous statistics requires the understanding of the semantics of each of the statistics, heuristic knowledge about the domain of application, and a general understanding of the discipline of statistics. Therefore, techniques for automated support for statistical information integration require the explicit use of heuristic knowledge and cannot be seen as a mere statistical problem and the implementation of a statistical technique. The full text of this paper is available in [1].

2. Statistical Support Model

Specific heterogeneity problems of statistics are population differences, differences in reported statistics, and classification differences. The focus of this paper is on classification differences. From human experts the following recipe was acquired ("weight matrix method"):

- Select the most reliable primary sources that contain the requested statistical variable for the requested populations.
- For each primary source find a source containing the primary source classification variable, the goal classification variable, and a variable that can be used as proxy for the requested statistic.
- Construct the weight matrix *W* and multiply primary source with it.

The statistical support model basically follows these steps. It includes an ontology of generic statistical terms and relations, and a composed process model with knowledgebases about (1) generic knowledge for solving classification differences, (2) metadata about available sources, and (3) domain knowledge represented as statistical models.

Example: German fisheries statistics use length as ship size indicator. Danish statistics use gross register tonnage (GRT). Suppose a table is requested to compare German catch with Danish data in GRT. No primary source is available, but a there is a German source that specifies total catch by ship length. Furthermore the German fleet register is available, giving length, GRT and engine power for each ship. Applying domain specific knowledge (catch is approximately proportional to engine power), a weight matrix can be computed from the fleet register.

3. Discussion

In the research reported in this paper, the focus was on heuristics and software support. The "weight matrix method" distilled from expertise of humans in the field takes a central place in the model. From examples that were studied, the weight matrix method appears to give reliable results. Model structure is set up in such a way that it allows easy extension with other methods. Current research is focused on extension of the system with statistical techniques to quantify the reliability of the results. Future research will create dedicated software for statistical techniques to further support integration of heterogeneous statistics.

References

[1] C.M. Jonker and D.Verwaart. Intelligent Support for Solving Classification Differences in Statistical Information Integration. *Proceedings of the Sixteenth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE 2003)*, LNAI 2718, Springer, 2003.

Organisational Change: Deliberation and Modification (extended abstract) [1]

Catholijn M. Jonker[a] Martijn Schut[a] Jan Treur[a,b]

[a] Vrije Universiteit Amsterdam
Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
URL: http://www.cs.vu.nl/~{jonker, schut, treur}

[b] Universiteit Utrecht, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

Within the field of Organisation Theory organisational structures regulating societal dynamics, and thus entailing organisational behaviour are studied; e.g., [2]. A particular area for which this field has become quite relevant is the area of virtual (Internet-supported) organisations. Supporting the design of virtual organisations based on information agents on the Internet asks for a dedicated organisation modelling approach. Within the area of Computational Organisation Theory and Artificial Intelligence, a number of organisation modelling approaches have been and are being developed to simulate and analyse dynamics within organisations; e.g., [3], [4], [1]. Some of these approaches explicitly focus on modelling organisational structure, abstracting from the detailed dynamics. Other approaches put less emphasis on organisational structure but focus on the dynamics in the sense of implementing and experimenting with simulation models. The approach as described in [1] is an example of an approach focussing on organisational structure, abstracting from the details of the dynamics.

Organisational change indicates that an organisation is changing its behaviour over time, a phenomenon that receives much attention in recent literature on Organisation Theory; e.g., [2], [3]. Organisational change is a process that allows an organisation to adapt its behaviour to changing

[1] In: Klusch, M., Omicini, A., Ossowski, S., and Laamanen, H. (eds.), Cooperative Information Agents VII, Proceedings of the Seventh International Workshop on Cooperative Information Agents, CIA 2003. Lecture Notes in Artificial Intelligence, vol. 2782, Springer Verlag, 2003, pp. 336 - 344.

environmental conditions. In virtual organisations such changes occur on a regular basis, and in fact may be part of the normal functioning of such an (evolving) organisation.

The initiative for changes of organisational structure usually lies within the organisation (in interaction with the environment). In organisations in human society, often the underlying decision process is embedded within the organisation in the form, e.g., of a director or management board supported by a strategic management department. In this sense, the process to obtain a changed organisation is itself part of the organised dynamics. This makes the organisational dynamics a reflective process. To model the dynamics of this reflective process in order to support the evolution of virtual, information agent-based organisations is the challenge addressed in this paper. As such, an example scenario is described, and an explanation of the manner in which strategic management for organisational change is modelled. A prototype implementation and simulation results are discussed.

References

[1] Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organi-sations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), IEEE Computer Society Press, pp. 128-135.

[4] Huczynski, A. and Buchanan, D. (1985). Organizational Behaviour, Prentice Hall

[7] Lomi, A., and Larsen, E.R. (2001). Dynamics of Organizations: Computational Modeling and Organization Theories, AAAI Press, Menlo Park.

[8] Prietula, M., Gasser, L., Carley, K. (1997). Simulating Organizations. MIT Press.

A Resource Based Framework for Planning and Replanning

Roman van der Krogt, Mathijs de Weerdt and Cees Witteveen

Parallel and Distributed Systems Group,
Delft University of Technology,
{r.p.j.vanderkrogt, m.m.deweerdt, c.witteveen}@cs.tudelft.nl

The full version of this paper has been accepted for the 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003).

Usually, a planning problem is specified using a description of (i) the *current* (or *initial*) *state* an agent is in, (ii) the set of *actions* the agent is capable to perform, and (iii) the *goals* the agent is aiming at. The planning *problem* then is to find the right sequence (or partial order) of actions leading the agent from the initial state to one of the states specified by the goals. Currently, there exists many systems and methods that try to tackle the planning problem. These systems, however, have some serious drawbacks.

First of all, almost all of these planning systems rely on the tacit assumption that planning problems always can be solved *off-line*: the goal and the initial state are assumed to remain unchanged. Due to the dynamical nature of most planning problems, however, for a large number of domains this assumption simply does not hold.

Secondly, most planning systems take for granted that a planning agent has to start from scratch. Often, however, this assumption is not realistic: agents are able to use results of their previous planning experiences, or knowledge given to them by a domain expert. The standard approach to planning seems to be just a *limiting case* of standard practice and usually needs to be generalized to include the adaptation of existing plans.

Fortunately, there are planning systems that at least partially deal with these issues. For example, the Systematic Plan Adaptor (SPA) [2] system meets the second objection by addressing plan adaptation. This *case-based planner* maintains a database of past problems and their solution plans, and chooses an appropriate starting plan whenever it faces a planning problem.

This plan then is modified to match the current goal and initial state requirements. Other systems focus on the replanning aspect to tackle the first problem. A system as GPG [1] finds the problems that exist in a plan (e.g. preconditions of actions that are not satisfied) and tries to replace parts of the plan such that these problems are solved.

This paper tries to overcome the problems mentioned by introducing a framework that brings together ideas from both planning and replanning approaches. It is based upon an existing logic-based framework for resource based planning [4]. We show that refinement strategies can be built on top of this framework to supply computational support for (re)planning. We consider such a unifying framework for planning and replanning to have (at least) the following benefits. First of all, it should offer a common platform to develop new heuristics and algorithms. Secondly, it should offer possibilities to compare the quality of competing (planning) algorithms.

This paper is organized as follows. First, we give a concise introduction to an existing resource-based planning approach [4]. Next, we extend this formalism and use it to deal with replanning and we introduce plan transformation operators that are able to modify resource-based plans. By assuming that an agent is able to use a *plan library*, these operators can be used to transform an initial (inadequate) plan into an adequate plan. We present a method to use existing planning techniques in this framework, and show that the FF-approach [3] and SPA [2] algorithm for plan adaptation can be conceived as special cases of this (re)planning as plan transformation approach.

References

[1] A. Gerevini and I. Serina. Fast plan adaptation through planning graphs: Local and systematic search techniques. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS-00)*, pages 112–121, 2000.

[2] S. Hanks and D. Weld. A domain-independent algorithm for plan adaptation. *Journal of AI Research*, 2:319–360, 1995.

[3] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of AI Research*, 14:253–302, 2001.

[4] J. Tonino, A. Bos, M. M. de Weerdt, and C. Witteveen. Plan coordination by revision in collective agent-based systems. *Artificial Intelligence*, 142(2):121–145, 2002.

Lino, the user-interface robot

B.J.A. Kröse [a] J.M. Porta [a] A.J.N. van Breemen [b]

K. Crucq [b] M. Nuttin [c] E. Demeester [c]

[a] Informatics Institute, University of Amsterdam, The Netherlands
[b] Philips Research, Eindhoven, The Netherlands
[c] Katholieke Universiteit Leuven, Leuven (Heverlee), Belgium

Abstract

This is an extended abstract of a paper which is presented at the European Symposium on Ambient Intelligence, Nov 3-4 2003, Eindhoven, the Netherlands.

Context

In the last years an increasing effort is spent in research on service and entertainment robots which operate in natural environments and interact with humans. The Sony AIBO is an example of a robot which is meant to play with children: it has a perceptual system (vision, auditory, tactile), plays soccer, and can learn its own behavior. NEC has developed "Papero", a *personal* robot which also is able to entertain the user but has more functionality: it serves as an interfacing with web-services and electronic equipment.

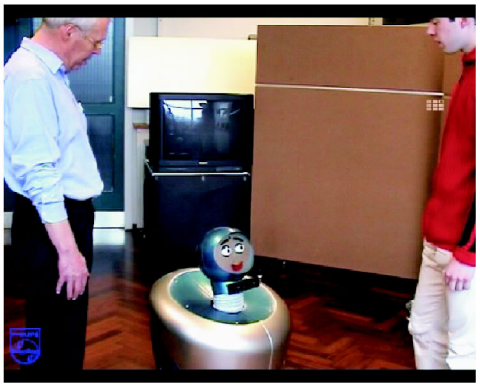


Figure 1: The robot Lino talking to people.

As a part of the European project "Ambience" we developed a domestic robot (see Figure 1). The robot must be some personification of the intelligent environment, and it must be able to show intelligent behavior, context awareness and natural

interaction. The robot exploits the intelligent environment to get information about the user intentions, preferences, etc. In the other way around, the human user must be able to have a natural interaction with the digital world by means of the robot.

Achievements

The current prototype shows research results from different areas.

Software Framework We have developed a module-based software framework, called the *Dynamic Module Library* that enables software development by multiple parties. A *module* has input and output ports, which can be connected (during runtime) to each other to exchange data.

User Awareness Module Lino is equipped with microphones and camera's. Humans can be localized and recognized by their speech and appearance.

Speech A simple dialogue management system has been devised which basically functions by transitions to different states. The viseme output of the speech synthesizer is used to control and synchronize the lip movement during speech output. This lip synchronization contributes a lot to a more lively appearance.

Emotion Engine To generate the appropriate facial expressions and body language we have developed an emotion engine. This emotion engine autonomously reasons on the emotional state of the robot and is based on the psychological model of Ortony, Clore and Collins (OCC-model).

Localization and Navigation The robot is able to navigate to any desired location in the house. It uses 'appearance-based' methods for localization. A hybrid architecture is used in the navigation module combining the planned and reactive behaviour.

High Level Reasoning Module In order for the robot to realize high level goals it must be capable of reasoning about the information it has about its world. We planned to use the Belief, Desires and Intention (BDI) architecture, but for the experiments we wrote scenario's in a language "Q" and incorporated this in an expert system (CLIPS).

Conclusions

Despite the complexity of the system, the team was able to deliver a working prototype. We believe that the developments in hardware technology (camera's, actuators, computing power) and software will enable to have some real humanoid interfaces in a couple of years. Robustness is the key word in future research, especially for the perception system. Adaptivity and fusion of multiple modalities are a prerequisite for this.

Event-coreference across Multiple Multi-lingual Sources in the MUMIS Project*

Jan Kuper[§], Horacio Saggion[†], Hamish Cunningham[†], Thierry Declerck[‡], Ed Hoenkamp[¶], Marco Puts[¶], Franciska de Jong[§], Yorick Wilks[†], Peter Wittenburg[♯]

[§]Department of Computer Science, University of Twente, The Netherlands
[†]Department of Computer Science, University of Sheffield, UK
[‡]DFKI GMbH, Saarbruecken, Germany
[¶]NICI, University of Nijmegen, The Netherlands
[♯]Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands
email: jankuper@cs.utwente.nl

This paper describes work done in the context of the MUMIS project, especially on multi-document information extraction. The MUMIS project aims at searching and indexing video material on the domain of soccer, based on information extracted from textual documents in various languages.

As a case study, the project examined the European Soccer Championship, which took place in 2000.

The main line of the process is as follows:

- *Information extraction* from individual texts in various languages (English, German, Dutch) by information extraction systems developed in Saarbruecken, Sheffield, Enschede. This information extraction is based on an ontology and lexicon created for the soccer domain, and delivers key events (such as corner, shot-on-goal, free-kick) together with the names of the players involved in the event, and the time the event took place during a soccer match.

- *Merging* the results from the various information extraction systems into a combined result. Information extracted from single documents turns out to be often incomplete, or even incorrect. The merging part of the project uses domain knowledge, expressed in constraints, to combine partial information elements into more complete knowledge.

*MUlti-Media Indexing and Searching (parlevink.cs.utwente.nl/projects/mumis), Funded by EC's 5th Framework HLT Programme under grant number IST-1999-10651

The novelty of the project is the merging component, and the presentation will emphasise on this component. In some greater detail, the merging component consists of the following steps:

- *Two document alignment* to decide which (partial) events extracted from two documents do stem from the same event in reality,
- *Multi-document alignment* combines these two-document alignments into larger structures of corresponding events, where each structure contains (partial) events which are assumed to belong together, i.e., which are about the same event in reality,
- *Unification* of events based on domain knowledge expressed in constraint rules. Erroneous elements of partial events are corrected, and empty slots are filled in. There are various types of rules used in this step: event internal rules, rules expressing possible combinations of events, rules based on the role of the teams during the corresponding moment in the match,
- *Ordering* events in the right temporal order. For example, a shot-on-goal and a goal will typically occur in that order, and not in the opposite order.

Evaluation results show a substantial increase of the quality of the extracted information.

References

This paper is a summary of papers published a.o. at EACL'03 and IJCAI'03:

H. Saggion, J. Kuper, H. Cunningham, T. Declerck, P. Wittenburg, M. Puts, E. Hoenkamp, F. de Jong, Y. Wilks, *Event-coreference across Multiple Multi-lingual Sources in the MUMIS project*, in: Conference Companion to the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03), Budapest, April 12–17, 2003, 239–242

J. Kuper, H. Saggion, H. Cunningham, T. Declerck, F. de Jong, D. Reidsma, Y. Wilks, P. Wittenburg, *Intelligent Multimedia Indexing and Retrieval through Multi-source Information Extraction and Merging*, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03), Acapulco, August 9–15, 2003 409–414

A Dialogue Game for Inconsistent and Biased Information

Henk-Jan Lebbink [abc] Cilia L.M. Witteman [b]
John-Jules Ch. Meyer [a]

[a] Institute of Information and Computing Sciences, Utrecht University
[b] Diagnostic Decision Making, Faculty of Social Sciences, University of Nijmegen
[c] Emotional Brain research and consultancy, Almere

henkjan@cs.uu.nl

Extended Abstract. In a paper by Lebbink *et al.* [4], a dialogue game is presented that describes coherent conversational sequences at the speech act level between agents with inconsistent and biased information. These types of information are represented with a bilattice structure, and, accordingly, the agent's cognitive state is represented by theories of multi-valued logic which are based on a bilattice structure.

In real life conversations, humans can state inconsistent or biased information without any difficulty. Inconsistent communicative acts are observed when the information content of different acts is contradictory or the content of single acts is inconsistent. An agent may have evidence e.g. to believe something, and, at the same time, as much evidence to believe the contrary. In this case, the agent's belief state is *inconsistent*. A belief state is called *biased* when agents have more evidence to believe than to disbelieve something. We address an analysis of these epistemic modalities in dialogue games for multi-agent systems.

In Beun [1], a dialogue game is described by : a) the agent's cognitive state as a set of propositions, b) dialogue rules that define applicable communicative acts based on the agent's cognitive state, and c) update rules that change the agent's cognitive state as a result of communicative acts. Our objective is to show that in our dialogue game inconsistent and biased information can be dealt with consistently. To do this, we introduce truth-values from a bilattice structure [3, 2] in the rules and the agents' cognitive states. Based on this bilattice, a multi-valued logic is defined to represent the agent's epistemic attitudes towards its world.

Bilattice structures describe truth-values that represent, apart from the classical true and false, unknown and inconsistent information states, as well as a continuum of truth-values representing biased information states. The information order $\leq_k$ states that e.g. true has more information than unknown and the truth-order $\leq_t$ states that e.g. true is more true than the inconsistent state. In Figure 1, the smallest possible bilattice is given. The cognitive state of the agent is formulated as a set of multi-valued (logical) theories, which, consequently, can cope with the mentioned epistemic modalities. In the resulting dialogue game, agents can deal with monotonic information addition that leads to possibly inconsistent cognitive states without belief revision.

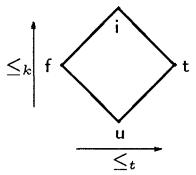


Figure 1: Bilattice 4

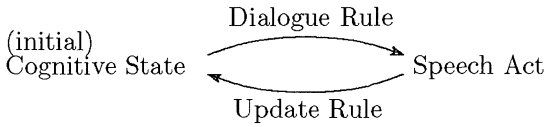


Figure 2: Dialogue cycle

A dialogue game defines a space of different dialogues, prescribed by a set of dialogue rules, a set of update rules and the agents' initial cognitive state. The dialogue game unfolds in a cycle (Figure 2). Three communicative acts are defined: a question from agent x to y which is read as *'May I accept to believe proposition p?'*, a grant which is read as *'You may accept to believe p'* and a denial of a question which is read as *'You may not accept to believe p'*. Eight mental constructs are defined that form the cognitive state of the agent. To give three examples: agent x's belief state (denoted B_x), the manifested ignorance state of another agent y as x is aware of (denoted $M_x I_y$), and the manifested ignorance state of agent x that y is aware of that agent x is aware of (denoted $M_x M_y I_x$).

The dialogue rule that defines the applicable question is given as an example. Assume $p{:}\theta$ is a propositional formula from a multi-valued logic with the reading *'proposition p has at least truth-value θ'*. An agent x is allowed to ask agent y a question in which x asks whether y believes $p{:}\theta$ to be the case. This question is applicable if: 1) $p{:}\theta \notin B_x$ stating that the proposition is not already believed by x (because an agent is not allowed to ask something that it already believes to be the case), 2) $p{:}\theta \notin M_x I_y$ stating that agent x is not aware that agent y is ignorant about the proposition (because an agent is not allowed to ask for propositions if it believes the other does not have an answer), and 3) $p{:}\theta \notin M_x M_y I_x$ stating that agent x is aware that agent y is aware that x is ignorant about the proposition (because an agent is not allowed to ask a question twice). If these three preconditions hold, agent x is allowed to pose the question.

We formally checked an example dialogue with inconsistent and biased information to be valid in our dialogue game consisting of seven dialogue rules and three update rules.

References

[1] R.J. Beun. On the Generation of Coherent Dialogue: A Computational Approach. *Pragmatics & Cognition*, 9(1):37–68, 2001.

[2] M. Fitting. Bilattices and the semantics of logic programming. *J. of Logic Programming*, 11:91–116, 1991.

[3] M.L. Ginsberg. Multivalued Logics: A Uniform Approach to Reasoning in Artificial Intelligence. *Comput. Intelligence*, 4:265–316, 1988.

[4] H-J. Lebbink, C.L.M. Witteman, and J-J.Ch. Meyer. Dialogue Games for Inconsistent and Biased Information. In W. vd Hoek, A. Lomuscio, E. de Vink, and M. Wooldridge, editors, *Workshop on Logic and Communication in Multi-Agent Systems (LCMAS'03)*, Eindhoven, Netherlands, June 29 2003.

Supervised locally linear embedding

Dick de Ridder[a], Olga Kouropteva[b], Oleg Okun[b],
Matti Pietikäinen[b] and Robert P.W. Duin[a]

[a]Pattern Recognition Group,
Dept. of Imaging Science and Technology,
Delft University of Technology,
Lorentzweg 1, 2628 CJ Delft, The Netherlands
http://www.ph.tn.tudelft.nl
[b]Machine Vision Group,
Infotech Oulu & Dept. of Electrical and Information
Engineering, P.O.Box 4500, FIN-90014 University of Oulu, Finland
http://www.ee.oulu.fi/mvg/mvg.php

Abstract

This work has been published before as: D. de Ridder, O. Kouropteva, O. Okun, M. Pietikäinen, and R.P.W. Duin, Supervised Locally Linear Embedding, in: O. Kaynak, E. Alpaydin, E. Oja, L. Xu (eds.), Artificial Neural Networks and Neural Information Processing, Lecture Notes in Computer Science, vol. 2714, Springer Verlag, Berlin, 2003, 333-341.

In many real-world classification problems, high-dimensional data sets are collected, e.g. from sensors. Often, the ideal decision boundary between different classes in such sets is highly nonlinear. A classifier should therefore have many degrees of freedom, and consequently a large number of parameters. As a result, training a classifier on such data sets is quite complicated: a large number of parameters has to be estimated using a limited number of samples. This is the well-known *curse of dimensionality*.

One can overcome this problem by first mapping the data to a high-dimensional space in which the classes become (approximately) linearly separable. Kernel-based techniques, such as support vector machines (SVMs), are typical examples of this approach. An alternative is to *lower* the data dimensionality, rather than increase it. Although it might seem information is lost, the reduction in the number of parameters one needs to estimate can result in better performance. Many linear methods for performing dimensionality reduction, such as principal component analysis (PCA) and linear discriminant analysis (LDA) are well-established in literature.

Here, a nonlinear dimensionality reduction method called locally linear embedding (LLE, [4]) is considered. The main assumption behind LLE is that the data set is sampled from a (possibly nonlinear) manifold, embedded in the high-dimensional space. This manifold is first approximated locally, by reconstructing each sample linearly from its K nearest neighbours. A second, global step then embeds the data by preserving these reconstructions as well as possible. LLE is an unsupervised, non-iterative method, which avoids the local minima prob-

lems plaguing many competing methods (e.g. those based on the EM algorithm). Some other advantages of LLE are that few parameters need to be set (selecting optimal values for these is discussed in [1, 3]) and that the local geometry of high-dimensional data is preserved in the embedded space.

To extend the concept of LLE to multiple manifolds, we propose a supervised variant of LLE, called SLLE. The objective changes from representing the manifolds as well as possible, to retaining class separability as well as possible. This can be achieved by distorting the distance matrix, on which LLE is calculated, such that distances between samples belonging to different classes are enlarged. If this enlargement is such that the K nearest neighbours of a sample will always be found in the same class [2], the resulting mapping will collapse each class into a single point (i.e. for a C-class problem, embedding in a $(C-1)$-dimensional space suffices). When the enlargement of the distances is less, a trade-off between embedding and class separation is achieved [1]. This may lead to better generalisation, at the cost of selecting an enlargement factor and an embedding dimensionality.

Extensive experiments show that simple classifiers, specifically the nearest mean classifier, trained on SLLE-mapped data can outperform other classifiers (k-nearest neighbour, linear and quadratic Bayes plugin classifiers) on the original data or data mapped by PCA, LDA, MDS or LLE. This is particularly true for high-dimensional data with a low intrinsic dimensionality, such as image data. In future work we intend to investigate possible speed-ups of the algorithm, as well as a way of automatically choosing the enlargement constant and embedding dimensionality.

Acknowledgements

The financial support of the Infotech Oulu graduate school is gratefully acknowledged. This work was partly sponsored by the Dutch Foundation for Applied Sciences (STW) under project number AIF.4997.

References

[1] D. de Ridder and R.P.W. Duin. Locally linear embedding for classification. Technical Report PH-2002-01, Pattern Recognition Group, Dept. of Imaging Science & Technology, Delft University of Technology, Delft, The Netherlands, 2002.

[2] O. Kouropteva, O. Okun, A. Hadid, M. Soriano, S. Marcos, and M. Pietikäinen. Beyond locally linear embedding algorithm. Technical Report MVG-01-2002, Machine Vision Group, University of Oulu, Finland, 2002.

[3] O. Kouropteva, O. Okun, and M. Pietikäinen. Selection of the optimal parameter value for the locally linear embedding algorithm. In *Proc. of the* 1^{st} *Int. Conf. on Fuzzy Systems and Knowledge Discovery, Singapore*, pages 359–363, 2002.

[4] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

An experience report on using DAML-S

Marta Sabou [a] Debbie Richards [b] Sander van Splunter [a]

[a] Dept. of Artificial Intelligence, Vrije Universiteit Amsterdam
[b] Division of I.C.S., Macquarie University, Sydney

Abstract

Though DAML-S is growing into a *de facto* standard for semantic web-service markup, we have only found few complete service descriptions and even less papers discussing technical issues about the markup process. We addressed this lack by (1) reporting on our experiences in describing a set of services, (2) concluding several limitations of the latest DAML-S version (v0.7) and (3) making our work accessible to the research community[1]. This paper was presented at the Workshop on E-Services and the Semantic Web (ESSW '03), The World Wide Web Conference. Budapest,2003.

1 Introduction

DAML-S is an initiative of the Semantic Web community to facilitate automatic discovery, invocation, composition, interoperation and monitoring of web-services (WSs) through their semantic description [1]. DAML-S is a DAML+OIL ontology divided into three sub-ontologies for specifying *what a service does?* (Profile), *how the service works?* (Process) and *how the service is implemented?* (Grounding). The grounding aligns the semantic specification with implementation details described using WSDL, the industry standard for web-service description.

DAML-S has generated a lot of interest through its promise to add semantics to web service descriptions. Despite that interest we were only able to find a small number of DAML-S descriptions of web services and most of these did not point to real services and were from the DAML-S community. Within the DAML-S coalition two complete, fictitious examples (on the DAML-S site) are provided. Several other projects use only certain parts of the DAML-S ontology, eg. match-making research tends to focus on the Profile ontology. Other researchers report on extending parts of DAML-S, e.g. by enriching the Process/Profile ontologies. Finally, some papers do mention use of complete DAML-S as is, but their purpose was to describe other research work therefore ignoring any details about their experiences with the language. Common to all the existing papers is that none of them describe the process of writing the DAML-S markup. We sought to fill this gap by providing a set of complete, real web service descriptions and sharing our modelling experiences.

[1] All services available at http://www.cs.vu.nl/~marta/services/

2 Using DAML-S

We marked up a set of web services which are used to build web-portals from semantically annotated bibliographic information. An agent based configuration service uses these descriptions to configure the suit of services depending on the characteristics of the input data. Based on our modelling experiences we have distilled some general observations about DAML-S.

We have concluded that DAML-S is superior to existing WS languages as it allows use of formally defined domain knowledge. It goes beyond syntactic description of a service by providing a semantic description. Semantics allow reasoning about a service and move us towards the ultimate goal of dynamic service discovery and usage. The other key strength of DAML-S is that it links to an industry standard, namely WSDL. In this way, it indeed fulfills its role as a link between the Semantic Web community and industry.

However, we also encountered a set of shortcomings.

A) Imprecise conceptual model. While it is commendable that DAML-S seeks to provide flexibility and thus has not fully defined a number of its concepts, this flexibility comes at the expense of clarity. The result of this imprecision is that DAML-S has an imprecise underlying conceptual model. We base this on the following facts. First, the three parts of DAML-S employ different metaphors to describe the same service: a program metaphor, an action metaphor and a view based on network endpoints. Second, several links exist between the conceptual models however they are often unclear. Our paper demonstrates that the link between the Profile and Process ontologies leads to inconsistencies in the final descriptions. Third, there is no clear correspondence of DAML-S concepts with software engineering (SE) concepts therefore it is often ambiguous how to model well-accepted software paradigms.

B) Mapping to WSDL limits DAML-S expressivity. We have experienced that the mapping to WSDL often limits the expressivity of DAML-S. Just by modelling a simple service we conflicted with two out of three basic assumptions that underlie the mapping. This forced us to revise our descriptions so that a grounding was possible at the expense of giving up specification of parametric polymorphism or an accurate specification of a complex internal structure. We feel that DAML-S is influenced too much by the actual grounding details.

C) Difficult to learn. One of our major comments (and worries) is that it was quite difficult to get started with writing DAML-S. The previously mentioned lack of conceptual model played a fair role in this. Other inhibiting factors were the limited tool support and the low number of provided examples.

We hope that both our observations and actual examples will contribute to the development and large scale usage of DAML-S and move our community closer to realizing the Semantic Web.

References

[1] DAML Services Coalition. DAML-S: Semantic Markup for Web Services. DAML-S v. 0.7 White Paper, October 2002.

Non-standard reasoning services for the debugging of description logic terminologies

Stefan Schlobach[a] Ronald Cornet [b]

[a] Language and Inference Technology, ILLC, UvA, NL
[b] Academic Medical Center, UvA, NL

Abstract

This is an extended abstract of: S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence IJCAI-03, Acapulco, Mexico, 2003. Morgan-Kaufmann Publishers.

Developing a terminology is a time-consuming and error-prone process. DICE, a terminology developed at the AMC in Amsterdam for the unambiguous and unified classification of patients in Intensive Care medicine, defines more than 2400 concepts and uses 45 relations. Let us illustrate some of the problems: in a first version of DICE a "brain" was incorrectly specified, among others, as a "central nervous-system" and "body-part" located in the head. This definition is contradictory as nervous-systems and body-parts are declared disjoint in DICE. Fortunately, current Description Logic reasoners, such as RACER, can detect this type of inconsistency and the knowledge engineer can identify the cause of the problem. Unfortunately, many other concepts are defined based on the erroneous definition of "brain" forcing each of them to be erroneous as well. In practice, DL reasoners provide lists of hundreds of unsatisfiable concepts for the DICE TBox and the debugging remains a jigsaw to be solved by human experts, with little additional explanation to support this process.

By *debugging* we understand the identification and elimination of modelling errors when detecting logical contradictions in a knowledge base. In our paper we focus on the former and define a number of new non-standard reasoning services to explain incoherences through *pinpointing*. Our experience with debugging DICE provides some hands-on examples for the problem at hand: take the contradictory definition of brains in the DICE anatomy specification. What information is useful for correcting the knowledge base? First, we have to identify the precise position of errors within a TBox; that is, we need a procedure to single out the axioms causing the contradiction.

Axiom Pinpointing The axioms for *Brain* and *CentralNervousSystem* form such a minimal incoherent subset of the DICE terminology. Formally, we introduce *minimal unsatisfiability-preserving sub-TBoxes* (abbreviated MUPS) and *minimal incoherence-preserving sub-TBoxes* (MIPS) as the smallest subsets of axioms of an incoherent terminology preserving unsatisfiability of a particular, respectively of

at least one unsatisfiable concept. Let us study a simple example: consider the following (incoherent) TBox $\mathcal{T}_1$, where A, B and C are primitive and $A_1, \ldots, A_7$ defined concept names:

$A_1 \doteq \neg A \sqcap A_2 \sqcap A_3$	(ax_1)	$A_2 \doteq A \sqcap A_4$	(ax_2)
$A_3 \doteq A_4 \sqcap A_5$	(ax_3)	$A_4 \doteq \forall s.B \sqcap C$	(ax_4)
$A_5 \doteq \exists s.\neg B$	(ax_5)	$A_6 \doteq A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$	(ax_6)
$A_7 \doteq A_4 \sqcap \exists s.\neg B$	(ax_7)		

The set of unsatisfiable concept names is $\{A_1, A_3, A_6, A_7\}$. Although this is still of manageable size, it hides crucial information, e.g., that unsatisfiability of A_1 depends on unsatisfiability of A_3 because of the contradictions between A_4 and A_5. *Minimal unsatisfiability preserving sub-TBox* of $\mathcal{T}_1$ and the concept A_1 are, for example, $\{\{ax_1, ax_2\}, \{ax_1, ax_3, ax_4, ax_5\}\}$. We get three *minimal incoherence preserving sub-TBoxes* $\{\{ax_1, ax_2\}, \{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$. It can easily be checked that each of the three incoherent TBoxes is indeed a MIPS as taking away a single axiom renders each of the three coherent. The first one signifies, for example, that the first two axioms are already contradictory without reference to any other axiom, which suggests a modelling error already in these two axioms.

Concept Pinpointing Having pinpointed the error-relevant axioms of a TBox we now have to highlight the elements of these definitions containing the faulty specification. An axiom, e.g., $Brain \dot{\sqsubseteq} BodyPart \sqcap NervousSystem$ points to the core of the erroneously modelled knowledge. For this purpose we define *generalised incoherence-preserving terminologies* (GIT) as sets of incoherent axioms, which are syntacticly related to the original axioms, more general and have minimal structural complexity. Three minimal sized GITs exist for our example TBox $\mathcal{T}_1$, where we only show the non-trivial axioms: $\{\{A_1 \dot{\sqsubseteq} \neg A \sqcap A_2, A_2 \dot{\sqsubseteq} A\}, \{A_3 \dot{\sqsubseteq} A_4 \sqcap A_5, A_4 \dot{\sqsubseteq} \forall s.B, A_5 \dot{\sqsubseteq} \exists.\neg B\}, \{A_7 \dot{\sqsubseteq} A_4 \sqcap \exists s.\neg B, A_4 \dot{\sqsubseteq} \forall s.B\}\}$. Given that $\{ax_4\}$ occurs in two MIPS it is most likely to cause incoherence of $\mathcal{T}_1$. Furthermore, we will have to take care of a contradiction on A in (ax_1) and (ax_2). Practical experience with DICE has shown that the reason for incoherence could be the erroneous use of concepts as well as their actual definition.

Evaluation Tableau-based algorithms for axiom pinpointing have been implemented in JAVA using RACER, which provides the set of unsatisfiable concepts. We evaluated the methods on both the anatomy fragment of DICE and the full DICE terminology. The fragment defines 529 concept names, 76 of which were unsatisfiable at first. There are 5 MIPS containing 2 axioms each, but no axiom occurs in all MIPS. However,the axiom defining "central nervous system" as a "nervous system" occurs in three MIPS. Note that the definition of "nervous system" is *not* contradictory but that its *use* is erroneous, e.g., the concept *brain* should be defined as a "part of" and not as a subsumee of the concept *nervous system*. For each of the 5 MIPS minimal size GITs have been calculated, and they point to the exact position of the logical incorrectness. For the MIPS related to the error described in the introduction the minimal size GIT is simply $Brain \dot{\sqsubseteq} NervousSystem \sqcap BodyPart$ which, with the information that *BodyPart* and *NervousSystem* are disjoint concepts, identifies the erroneous specification.

Automated Negotiation and Bundling of Information Goods

Koye Somefun [a], Enrico Gerding [a], Sander Bohte [a], Han La Poutré [ab]

[a] CWI, Centre for Mathematics and Computer Science, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
[b] School of Technology Management, Eindhoven University of Technology,P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Extended Abstract[1]

Personalization of information goods becomes more and more a key component of a successful electronic business strategy. The challenge is to develop systems that can deliver a high level of personalization combined with a high adaptability to changing circumstances. We developed a system which can attain these properties through negotiation of information goods. Autonomous "software agents" perform the negotiation on behalf of the users of the system. The system is applied to subscriptions on news items, with subscriptions for a fixed and typically short time interval, e.g., a single day.

We consider the novel approach of selling bundles of news items through a system that allows for bargaining over multiple issues, viz. the price, the bundle content and the quality of service of the delivered goods. A bundle content defines which types of news categories the bundle contains, e.g. religion, culture, and politics. Within a category the system distinguishes between two quality of service levels: low quality for headlines only and high quality for the complete news article. A fixed price (p_f) is paid for receiving the bundle content with the specified quality of service. Moreover, a variable price (p_v) is additionally charged for each article if the customer chooses to read the full content and the category is of low quality of service.

Bargaining and Fairness. A customer bargains with the seller by exchanging offers and counter offers. An offer specifies the fixed price, the variable price (uniform for all low quality of service categories), the bundle content, and also the desired quality of service of the information for each category separately. To accelerate the negotiation process, customers can initiate concurrent negotiation threads for the same bundle with differences in the quality of the delivered bundles. The seller responds by varying the fixed and variable price. Bargaining continues until an agreement is reached or one of the bargainers terminates the process.

A possible drawback of bargaining is that two customers may end up paying a substantially different price for very similar bundles. Customers may perceive this as unfair. This is an important concern for the seller, since customers may become

[1] See [1] for the full paper.

dissatisfied or stop using the system altogether. In the system "fair" negotiations are ensured by allowing all customers to have the same opportunities within a certain time frame. The actual bargaining process is therefore both one-to-one since different customers can end up with personalized deals, and at the same time one-to-many partly due to the fairness constraint for which the seller must use the same strategy with all customers.

Decomposition of Negotiation Strategies. The customer agents and seller agent support various bargaining strategies to do the actual bargaining. These strategies make use of the notion of a utility function to represent the bargainers' preferences for the various issues. We explicitly decompose strategies in our system into (one-to-many) concession strategies and (one-to-one) "Pareto search" strategies.

Concession strategies determine what the desired utility level of an offer will be given a particular sequence of offers and counter offers. Algorithms that implement Pareto search strategies determine, given a particular utility level and a particular history of offers and counter offers, what the multi-issue offer will be, i.e., the fixed price p_f and the variable price p_v.

Pareto search strategies aim at reaching agreement as soon as the respective "concession" strategies permit this. Therefore it may be good for both parties to use a strategy which approximates Pareto-efficient deals. In more economic terms a negotiated deal is called Pareto efficient if it is impossible to obtain a better deal without making one of the bargainers worse off. From a system design perspective Pareto efficiency is clearly desirable.

In the full paper [1] we introduce two classes of Pareto search strategies: *orthogonal* and *orthogonal+derivative follower*. We show via computer experiments that if the customer agents use the orthogonal search strategy and the seller agent uses the orthogonal+derivative follower then the bargaining outcome will closely approximate a Pareto efficient solution given a wide variety of concession strategies. Although a customer is free to select other bargaining strategies, it is actually in the best interest of the customers to have their agents use the specified search strategies. We refer to the full paper for a detailed description of the system[1].

References

[1] Koye Somefun, Enrico Gerding, Sander Bohte, and Han La Poutré. Automated negotiation and bundling of information goods. In *Proceedings of the AAMAS-03 Workshop on Agent-Mediated Electronic Commerce V (AMEC V), Melbourne, Australia*, 2003.

Integrity and Change in Modular Ontologies *

Heiner Stuckenschmidt, Michel Klein
Vrije Universiteit Amsterdam
{heiner, michel.klein}@cs.vu.nl

May 19, 2003

Currently, research in the area of the semantic web is in a state where ontologies are ready to be applied in real applications such as semantic web portals, information retrieval or information integration. In order to lower the effort of building ontology-based applications, there is a clear need for a representational and computational infrastructure in terms of general purpose tools for building, storing and accessing ontologies. A number of such tools have been developed, i.e. ontology editors, reasoning systems and more recently storage and query systems. Most of these tools, however, treat ontologies as monolithic entities and provide little support for specifying, storing and accessing ontologies in a modular manner.

Why Modularization ?

There are many reasons for thinking about ontology modularization. Our work is mainly driven by three arguments. These also bias the solution we propose, as it is aimed at improving the current situation with respect to the following aspects.

Distributed Systems: In highly distributed systems such as the semantic web, modularity naturally exists in terms of physical location. Providing interfaces and mechanisms for connecting these natural modules is a prerequisite for easy maintenance.

Large Ontologies: Modularization also helps to manage very large ontologies we find for example in medicine or biology. Here modularity helps to maintain and reuse parts of the ontology as smaller modules are easier to handle than the complete ontology.

Efficient Reasoning: A specific problem that occurs in the case of distributed and large models is the problem of efficient reasoning. The introduction of modules with local semantics and clear interfaces will help to develop efficient reasoning methods.

*This is a summary of a paper accepted for the International Joint Conference on Artificial Intelligence IJCAI'03, August 2003, Acapulco, Mexico.

Our Approach

In the following, we describe our approach to ontology modularization on an abstract level. We emphasize the main design decisions and motivate them on the basis of the requirements defined above. The technical details of the approach will be given in the following sections.

View-Based Mappings: We adopt the approach of view-based information integration. In particular, ontology modules are connected by conjunctive queries. This way of connecting modules is more expressive than simple one-to-one mappings between concept names but less expressive than the logical language used to describe concepts. We decide to sacrifice a higher expressiveness for the sake of conceptual simplicity and desirable semantic properties such as independence of the ontology langauge used.

Compilation of Implied Knowledge: In order to make local reasoning independent from other modules, we use a knowledge compilation approach. The idea is to compute the result of each mapping query off-line and add the result as an axiom to the ontology module using the result. During reasoning, these axioms replace the query thus enabling local reasoning.

Change Detection and Automatic Update: Once a query has been compiled, the correctness of reasoning can only be guaranteed as long as the concept hierarchy of the queried ontology module does not change. In order to decide whether the compiled axiom is still valid, we propose a change detection mechanism that is based on a taxonomy of ontological changes and their impact of the concept hierarchy.

Key Results

We present two main contribution towards a better understanding and management of dependencies in the light of changes to an ontology.

- We present a formal model for describing dependencies between different ontologies. We propose conjunctive queries for defining concept using elements from another ontology and presented a model-based semantics in the spirit of distributed description logics that provides us with a notion of logical consequence across different ontologies. This clear semantic account of dependence makes it possible to study the impact of changes on a semantic level.

- We describe a method for detecting changes in an ontology and for assessing their impact. The main feature of this method is the derivation of conceptual changes from purely syntactic criteria. These conceptual changes in turn provide input for a semantical analysis of the effect on dependent ontologies, in particular on the validity of implied subsumption relations.

Consistency Techniques for Interprocedural Test Data Generation

Nguyen Tran Sy Yves Deville

Université catholique de Louvain
Place Saint-Barbe 2
B-1348 Louvain-la-Neuve, Belgium
{tsn,yde}@info.ucl.ac.be

Abstract

This paper[1] presents a novel approach for automated test data generation of imperative programs containing *integer*, *boolean* and/or *float* variables. It extends our previous work [1] to programs with procedure calls and arrays. A test program (with procedure calls) is represented by an Interprocedural Control Flow Graph (ICFG). The classical testing criteria (statement, branch, and path coverage), widely used in unit testing, are extended to the ICFG. For path coverage, the specified path is transformed into a *path constraint*. Our previous consistency techniques, the core idea behind the solving of path constraints, have been extended to handle procedural calls and operations with arrays. For statement (and branch) coverage, paths reaching the specified node or branch are dynamically constructed. The search for suitable paths is guided by the interprocedural control dependences of the program. The search is also pruned by a new specialized consistency filter. Finally, test data are generated by the application of the proposed path coverage algorithm. A prototype has been implemented. Experiments show the feasibility of the approach.

Keywords software testing, test data generation, procedures, arrays, constraint satisfaction, consistency

Introduction Structural testing techniques are usually concerned with the use of the control-flow a program to guide the generation of test data. The control-flow, in turn, is represented by a Control Flow Graph (CFG). To adequately test the program at the structural level, we must consider structural elements (nodes, branches, or paths) of the CFG for coverage. For example, *statement coverage* requires developing test cases to execute certain nodes of the CFG. Similarly, *branch coverage* requires test cases to traverse certain branches, and *path coverage* requires test cases to execute certain paths. Structural testing thus includes (1) choice of a criterion (statement, branch or path), (2) identification of a set of nodes, branches or paths, and (3) generation of test data for each element of this set. The automation of the last phase is a vital challenge in software testing.

[1]This paper has been published in the Proceedings of the Joint 9th European Software Engineering Conference and 11th ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'03), Helsinki , Finland, 2003

Existing approaches Classical testing approaches can be classified into the following categories. *Random* test data generation consists in trying test data generated randomly until an element is executed. *Symbolic evaluation* consists in replacing input variables by symbolic values, and then symbolically evaluates the statements along a path. It is however limited in handling arrays and procedure calls. *Program execution based* (or *dynamic*) approaches start by executing the program with an arbitrary test input. This input is then iteratively refined, by execution of the program, to obtain a final input, executing a path, a branch, or a statement. Although dynamic approaches are powerful in handling arrays and dynamic data structures, it may require a great number of executions of the program. Other approaches, based on *genetic algorithms* or *Constraint Logic Programming*, have also been proposed.

In the following problem statement, an Interprocedural Control Flow Graph (ICFG) is a classical representation of programs.

Problem statement *Given a node n, a branch b or a path p of the ICFG associated with a test procedure P (possibly with procedure calls), generate a test input i such that P when executed on i will cause n, b or p to be traversed.*

We propose a novel consistency-based approach for interprocedural test data generation. Statement, branch and path coverage criteria are all handled. Path coverage is the core of our approach. It includes the following steps. (1) A path constraint is derived from a specified path of the ICFG. Such a constraint can involve operations with arrays. (2) The path constraint is solved by a new specialized interval-arithmetic-based constraint solver extended to handle constraints invol ving arrays. (3) A test case is extracted from the interval solutions.

For statement (and branch) coverage, paths reaching the specified node or branch are dynamically constructed. Our algorithm for path coverage is then applied on these paths to generate test data.

Contribution The main contribution of the paper is a novel approach (based on consistency techniques), which is an extension of our previous paper [1] to generate test data for numeric programs (programs with integer, boolean and float variables) with procedure calls and arrays. This approach handles *branch*, *statement* and *path* coverage criteria. Specific technical contributions of the paper include the following. (1) A new method to obtain a path constraint directly from a path's traversal. (2) Two mechanisms for passing parameters (pass-by-value and pass-by-reference) in procedure calls are handled. (3) An improvement on our previous consistency techniques to tackle specific constraints involving arrays. (4) The proposed interval constraints solver integrates integers, reals, and booleans, as well as the logical operators AND, OR, NOT. (5) For statement and branch coverage, interprocedural control dependences are used during the search process, when the test procedure contains procedure calls.

References

[1] Nguyen Tran Sy and Yves Deville. Automatic test data generation for programs with integer and float variables. In *16th IEEE International Conference on Automated Software Engineering(ASE01)*, November 2001.

Production Compilation: A Simple Mechanism to Model Complex Skill Acquisition

Niels Taatgen[a] Frank Lee[b]

[a] University of Groningen, Grote Kruisstraat 2/1, Groningen and Carnegie Mellon University, Pittsburgh, PA, USA

[b] Drexel University, Philadelphia, PA, USA

Abstract

In this article [1] we describe *production compilation*, a mechanism for modeling skill acquisition that builds on the theories forwarded by Anderson and Newell and Rosenbloom. Production compilation has been developed within the ACT-R [2] cognitive architecture and consists of combining and specializing task-independent procedures into task-specific procedures. We provide an example of this process by developing and describing a model learning in a simulated air-traffic controller task.

1. The Goals of Cognitive Modeling

The goal of research in cognitive modeling is to not only simulate intelligent behavior on a machine, but also to do this in such a way that the simulation is as close as possible to human behavior. As such the research serves two purposes: the development of a theory of cognition, and a basis for machine intelligence.

The present article describes a computer simulation of learning a complex task: Air Traffic Control. In this study a simplified Air Traffic Control task is used [3], which is nevertheless complex enough to be interesting. The basis for the simulation is the ACT-R cognitive architecture [2]. A cognitive architecture provides a theoretical basis for cognitive models, and also serves as an implementation platform. ACT-R uses production rules an declarative facts to represent knowledge, and has the appropriate mechanisms to process information and to learn new knowledge. One mechanism that is of particular interest is a recent addition to the theory: *production compilation.* Production compilation is a mechanism that adds new production rules to the system, by instantiating and combining rules into new rules. This mechanism is very well suited to model learning a new task from instructions.

2. Skill acquisition

Human behavior on a new task is characterized by several phases. Initially, people are very slow, make many errors, and have to go through several thinking steps. They have to focus their full attention on the task, and are not able to do anything else at the same time. Through learning, this gradually changes: people get faster, make fewer errors, and eventually are able to do the task concurrently with other things (think about the prototypical "talk to the passenger while you drive" example).

3. The model

The model starts with the instructions to the task in declarative memory. This means that it has to interpret these instructions while doing the task, which is slow, and prone to errors because instructions can be incomplete or forgotten. The production compilation process gradually converts these instructions and experiences with the task into rules. This enables the model to be much faster, and improve its performance. The results of the model are compared to behavior of human participants, and it turns out there is a good match at the global performance level, at the level of sub-tasks within the main task, and even at the level of individual keystrokes. Actual or potential applications of this research include the evaluation of user interfaces, the design of systems that support learning and the building of user models.

References

[1] N.A. Taatgen and F.J. Lee. Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors*, 45(1): 61-76, 2003.

[2] J.R. Anderson, D. Bothell, M.D. Byrne, and C. Lebiere. An integrated theory of the mind. http://act-r.psy.cmu.edu/papers/403/IntegratedTheory.pdf

[3] P.L. Ackerman. Determinants of individual differences during *skill acquisition: Cognitive abilities and information processing. Journal of Experimental Psychology: General*, 117: 188-318, 1988.

A Variational EM Algorithm for Large-Scale Mixture Modeling

J.J. Verbeek N. Vlassis J.R.J. Nunnink

University of Amsterdam, Faculty of Science, Informatics Institute
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Introduction Mixture densities constitute a rich family of models that can be used in several data mining and machine learning applications, for instance, clustering. Although practical algorithms exist for learning such models from data, these algorithms typically do not scale very well with large datasets. In most cases the time complexity grows at least linear with the number of data points. Our approach, which builds on previous work by other authors, offers an acceleration of the EM algorithm for Gaussian mixtures by pre-computing and storing sufficient statistics of the data in the nodes of a kd-tree. Contrary to other works, we obtain algorithms that strictly increase a lower bound on the data log-likelihood in every learning step. Experimental results illustrate the validity of our approach.

Recently a speedup of the EM algorithm for Gaussian mixtures based on analyzing large chunks of data at once to save distance computations was proposed [1]. The idea is to use geometrical reasoning to determine that for chunks of data the posterior on mixture components hardly varies in the chunk of data. Cached sufficient statistics can then be used to perform the update step of the algorithms. In the full paper discussed here [3], we employ a variational EM approach and show how such speedups for the EM algorithm can be guaranteed to increase a lower bound on the data log-likelihood in each step. We also derive closed form and efficiently computable optimal assignments of chunks of data to mixture components. The variational EM framework allows for arbitrary partitionings, where existing techniques were forced to use relatively fine partitionings of the data. Thus the proposed framework allows more freedom in designing new speedup algorithms.

The variational EM algorithm Maximization of the data log-likelihood $\mathcal{L}(\theta)$ can be efficiently carried out by the EM algorithm. In this work we consider a variational generalization of EM [2] which allows for useful extensions. The variational EM algorithm performs iterative maximization of a lower bound of the data log-likelihood. In our case, the bound $\mathcal{F}$ is a function of the current mixture parameters θ and a soft assignment of the data points to the mixture components.

The normal EM algorithm to train mixtures sets in each E step the soft assignment equal to the posteriors $p(s|x_i)$, i.e. the probability that data point i belongs to mixture component s according to the mixture model with the current parameters setting. However, other assignments of the data points to the mixture components are also possible. Provided that $\mathcal{F}$ increases in each step, the

algorithm will optimize a lower-bound on the log-likelihood. The tightness of the bound depends on how close the used assignments are to the true posteriors.

The idea behind our fast EM algorithm is to use equal assignments for chunks of data points that are nearby in the input space. Consider a partitioning $\mathcal{A}$ of the data space into a collection of non-overlapping cells $\{A_1, \ldots, A_m\}$, such that each point in the data set belongs to a single cell. For all points in a cell $A \in \mathcal{A}$ we use the same assignment to the mixture components. We can compute the optimal 'shared' assignment easily from few sufficient statistics of the cell.

Note that, whatever partitioning we choose, the fast EM algorithm (which interacts with the data only through the cached statistics of chunks of data) is always a convergent algorithm that strictly increases in each step a lower bound on data log-likelihood. In the limit, if we partition all data points into separate cells, the algorithm will converge to a local maximum of the data log-likelihood.

Conclusions We present a variational EM algorithm that can be used to speed-up large-scale applications of EM to learn Mixtures of Gaussians. Our contributions over similar existing techniques are two-fold. Firstly, we can show directly that the algorithm maximizes a lower bound on data log-likelihood and that the bound becomes tighter if we use finer partitionings. The algorithm finds mixture configurations near local optima of the log-likelihood surface which are comparable with those found by the regular EM algorithm, but considerably faster. Secondly, because we have a convergent algorithm that maximizes a lower bound on data log-likelihood we can use any partitioning of the data. This allows us to use partitionings where the true posteriors differ heavily in each part, which might be needed when working on very large data sets and only limited computational resources are available. Comparing our work with [1], we conclude that with the same computational effort we can use the optimal shared assignments instead of the posterior at the node centroid. Furthermore, we obtained a provably convergent algorithm and have the freedom to use arbitrary partitionings of the data.

References

[1] A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In M. Kearns and D. Cohn, editors, *Advances in Neural Information Processing Systems*, pages 543–549. Morgan Kaufman, 1999.

[2] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer, 1998.

[3] J.J. Verbeek, N. Vlassis, and J.R.J. Nunnink. A variational EM algorithm for large-scale mixture modeling. In *Proc. 8th Ann. Conf. of the Advanced School for Computing and Imaging (ASCI 2003)*, Het Heijderbos, Heijen, 2003.

Iterated extended Kalman smoothing with Expectation-Propagation

Alexander Ypma[a] Tom Heskes[a]

[a] SNN Nijmegen, Geert Grooteplein 21, 6525 EZ, KU Nijmegen
Email: {ypma, tom}@mbfys.kun.nl

Abstract. We formulate extended Kalman smoothing in an expectation-propagation (EP) framework. The approximation involved (a local linearization) can be looked upon as a 'collapse' of a nongaussian belief state onto a Gaussian form. This formulation allows us to come up with better approximations to the belief states, since we can iterate the algorithm until no further refinement of the beliefs is obtained. Compared to the standard extended Kalman smoother, we linearize around the mode of the actual two-slice belief state instead of the predicted mean of the one-slice belief.

The full version of this paper will appear in the proceedings of the IEEE Int. Workshop on Neural Networks for Signal Processing, September 17-19 2003 at Toulouse (France). More information can be found on www.mbfys.kun.nl/~ypma/papers/list_of_papers.html

Model. In this paper we consider the dynamic systems where we have nonlinearities in the state- and observation equations,

$$\begin{aligned} x_t &= f(x_{t-1}) + v_t, \quad v_t \sim \mathcal{N}(0, \Gamma) \\ y_t &= g(x_t) + w_t, \quad w_t \sim \mathcal{N}(0, \Sigma) \end{aligned} \tag{1}$$

with conditional distributions

$$p(x_t|x_{t-1}) \sim \mathcal{N}(f(x_{t-1}), \Gamma), \quad p(y_t|x_t) \sim \mathcal{N}(g(x_t), \Sigma) \tag{2}$$

where $f(\cdot)$ and $g(\cdot)$ are (known) nonlinear functions, see figure 1. In the familiar Kalman filter and smoother all functions are assumed linear and so-called forward and backward messages (which serve as intermediate steps for computing the belief state at each time) can be computed exactly.

In the nonlinear model, the forward and backward messages cannot be computed exactly any more one has to resort to approximations (like the *extended* and the *unscented* Kalman filter). In this paper we propose an iterated Extended Kalman smoother.

Iterated smoothing with EP. One can express inference in a graphical model as a sequence of multiplications and a summation (an integral) of local factors and messages (which is equivalent to belief propagation in a graphical model). In the dynamic Bayesian network considered here (figure 1), no loops occur, but the inability to do an exact summation and to represent the nongaussian one-slice belief prohibits exact inference. The exact summation can however be replaced in

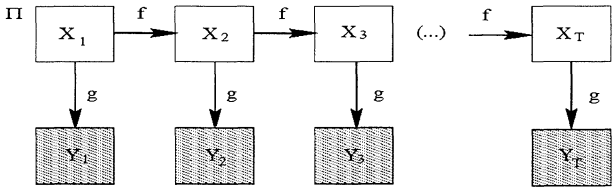


Figure 1: Nonlinear dynamical system. All nodes are continuous-valued, and f and g are arbitrary nonlinear functions. Shaded nodes are observed. π denotes the prior distribution on X. Time progresses from left to right.

the former procedure by a 'collapse' step, where one-slice beliefs are approximated by Gaussians. The exact posterior belief is then represented by a Gaussian that matches the first two moments of the exact posterior. These moments however cannot be found analytically, again because of the nonlinearities.

Algorithm for iterated extended Kalman smoothing. Therefore we propose an *approximate* moment match by linearizing the nonlinear argument of the non-Gaussian two-slice potential using a Taylor approximation. The moments of either $\hat{q}_t(x_t)$ (forward pass) or $\hat{q}_t(x_{t-1})$ (backward pass) are given by

$$\text{mom}(i,\tau) = \frac{1}{c}\int\int \alpha_{t-1}(x_{t-1}) \cdot \Psi_{t-1,t} \cdot \beta_t(x_t) \cdot G^i(x_\tau) dx_{t-1} dx_t \tag{3}$$

where $\text{mom}(i,\tau)$ is the ith moment of $\hat{q}_\tau(x_\tau)$, $i = 1,2$, $\tau = t-1, t$ and $G^i(x_\tau)$ is x_τ for $i = 1$ and $(x_\tau - m_\tau)^2$ for $i = 2$. We now 'collapse' the (non-Gaussian) posterior two-slice belief $q_t(x_{t-1}, x_t)$ to a Gaussian. We use a Laplace approximation of $F(\mathbf{x}_t)$ and $G(\mathbf{x}_t)$ around the extremum $\mathbf{x}_t^*$ of $F(\mathbf{x}_t)$ to arrive at the approximation

$$\exp\{F(\mathbf{x}_t)\} \approx \exp\{Q(\mathbf{x}_t)\} \sim \mathcal{N}(\mathbf{x}_t; \mathbf{x}_t^*, -(F'')^{-1}(\mathbf{x}_t^*)) \tag{4}$$

From this approximate two-slice belief, the approximate one-slice beliefs can be derived by marginalization,

$$\hat{q}_t(x_{t-1}) = \frac{1}{c_2}\int \exp\{Q(\mathbf{x}_t)\} dx_t \tag{5}$$

$$\hat{q}_t(x_t) = \frac{1}{c_2}\int \exp\{Q(\mathbf{x}_t)\} dx_{t-1} \tag{6}$$

Results. In initial experiments with a one-dimensional nonlinear dynamical system we found that our method improves over the extended Kalman filter and performs comparable to the unscented Kalman filter, whereas only second-order approximations are being made. The EP-formulation in principle allows for incorporation of higher-order approximations, possibly leading to further improvements.

Multi-Scale Switching Linear Dynamical Systems

Onno Zoeter Tom Heskes

SNN, University of Nijmegen
Geert Grooteplein 21, 6525 EZ Nijmegen, The Netherlands
{*orzoeter, tom*}*@snn.kun.nl*

Abstract

In [1] we introduce a class of models that is particularly useful for analyzing time-series in which different regimes at different level of detail can be identified.

In many real-world time-series the underlying system can be seen to operate in different *regimes*. That is, in some periods the system behaves according to different transition models or may be observed with different characteristics. In [1] we introduce a procedure to work with such systems in an efficient and flexible way. In particular if such systems follow a large number of regimes and these regimes can be seen to form a hierarchy. For instance if a system has several operating modes and within every mode several possible failures may occur (See Fig. 1). Instead of working with one complex model that works with all possible combinations of operating modes and failures, our setup exploits the hierarchical nature of such a system.

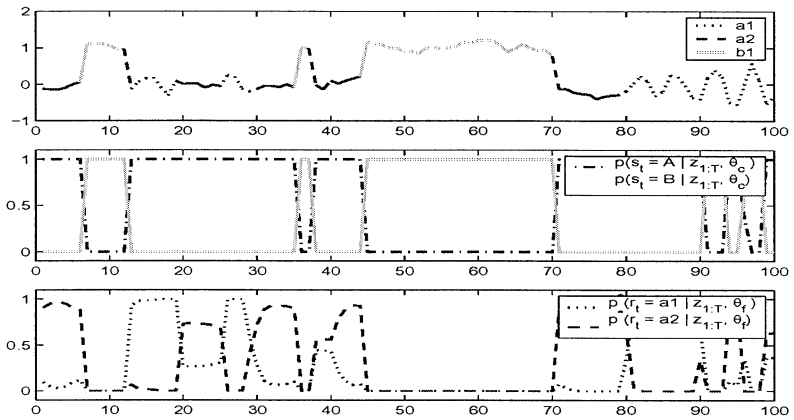


Figure 1: Artificial data (top), posterior probabilities over coarse regimes $\{A, B\}$ (middle) and posterior probabilities over refinements of regime A; $\{a_1, a_2\}$ (bottom). Note that the posterior probabilities of a_1 and a_2 add up to that of A.

In the multi-scale setup every level of coarseness has its own underlying generative model. If these models would be used independently to infer the state and

regime of the system, the outcomes would in principle be unrelated. To enforce agreement between different levels we follow a top-down approach. The state and regime posteriors are first inferred at the coarsest level. Then these outcomes are fixed, and used as constraints in the inference procedure for the finer level model. These constraints take the following form: if at the coarse level the posterior probability of being in a particular regime, say A, is 0.4, we enforce at the fine level that the posterior probabilities for refinements of A, say $a_1, \ldots, a_m$, sum to 0.4 (See Fig. 1). *Jeffrey's rule of conditioning* is used to enforce this agreement.

The underlying models we use are *Switching Linear Dynamical Systems (SLDS)*. These systems are powerful, but notorious for the fact that memory and computational requirements for exact inference grow exponentially with the number of observations. We describe a useful approximate inference algorithm for an SLDS that computes approximate posteriors over one and two time-slices.

The strictest requirements we might want to impose share the same exponential increase as inference itself. This is easy to see. If we denote the discrete regime indicators for T time-slices by $s_{1:T}$ at the coarsest level and by $r_{1:T}$ at the finer level the agreement requirement becomes

$$\sum_{r_1 \in \mathrm{ch}(s_1)} \cdots \sum_{r_T \in \mathrm{ch}(s_T)} p(r_{1:T}|\mathbf{z}_{1:T}, \boldsymbol{\theta}_\mathrm{f}) = p(s_{1:T}|\mathbf{z}_{1:T}, \boldsymbol{\theta}_\mathrm{c}) \ ,$$

where ch(.) is the parent-child relationship that defines the hierarchy in regimes, $\mathbf{z}_{1:T}$ are the observations and $\boldsymbol{\theta}_\mathrm{c}$ and $\boldsymbol{\theta}_\mathrm{f}$ are the parameters in the coarse and fine level models respectively. If there are M_c different coarse regimes there are M_c^T different regime histories we have to consider at the coarse level, and hence there are M_c^T constraints we have to enforce.

The approximation philosophy used for general inference can also be applied to enforce constraints. The inference algorithm changes slightly and the energy function that it tries to minimize changes accordingly.

To truly benefit from the hierarchy we want to be able to refine coarse regimes independently. We present the relatively mild restrictions that have to be put on a general SLDS to allow a fine level model to be treated as a collection of independent smaller models, one for every regime from the previous level.

The setup can then be used in demanding monitoring tasks: if certain regimes are very improbable, we can choose to not refine it (thereby eliminating the inference costs for its entire subtree) and only explore the modes of interest. Also, since MAP parameters estimation can be shown to share the same factorization, we hope this method paves the way for a greedy model selection algorithm that determines the required number of regimes.

References

[1] Onno Zoeter and Tom Heskes. Multi-scale switching linear dynamical systems. In *Proceedings of ICANN 2003*. Springer-Verlag, 2003.

Part 3

Demonstrations

3APL Platform

E.C. ten Hoeve, M. Dastani, F. Dignum, J.-J. Meyer

Institute of Information and Computing Sciences

Utrecht University

Abstract

The 3APL Platform is the first distributed multiagent hosting environment for cognitive agents. We demonstrate a Java implementation of the 3APL Platform developed at the Institute of Information and Computing Sciences at Utrecht University. More information on this project can be found at http://www.cs.uu.nl/3apl/thesis/tenhoeve/tenhoeve.html. The demonstration gives an overview of the functionalities of the platform. In particular, we will show how 3APL multiagent projects consisting of several interacting cognitive agents can be implemented and executed on this platform. Various aspects of the 3APL Platform such as its distribution across different machines, agent communication, agent management system, and graphical user interface will be demonstrated.

1. 3APL Platform

In recent years agent platforms, such as JADE [4], JACK [5], and ZEUS [6], have been created to develop and execute multiagent systems. Although these platforms provide communication facilities as well as specific tools and components such as agent management systems and directory facilitators to register agents and their services, they do not support direct development and execution of *cognitive* agents. The 3APL platform is the first platform that supports easy and direct implementation and execution of cognitive agents. The platform can be distributed across different machines connected in a network. On this platform, cognitive agents can be implemented directly in terms of mental attitudes such as beliefs, basic actions, goals, and reasoning rules [1]. The specifications of FIPA on agent platforms [2] were followed as a guideline. For example, the 3APL platform has an Agent Management System (AMS), which maintains a list of hosted agents on the platform. The AMS handles the lifecycle, i.e. when agents are created their identities are added to the list of available agents; agent identities remain on this list until the agents are removed from the platform.

An important feature of the 3APL platform is that it provides a message transport layer that supports agent communication when the platform is distributed across different machines that are connected in a network. Moreover, the 3APL platform is FIPA compliant in the sense that agents running on this platform can in principle communicate with agents that run on a different FIPA compliant platforms such as JADE. The 3APL platform has a graphical user interface that allows the observation of agent configurations and their changes at run-time. It also allows observing agent interactions and the messages they pass to each other. The GUI supports also easy addition and deletion of agents at run-time by providing template agents and a syntax-highlighting editor. The list of available agents, maintained by the AMS, is presented as a list of trees. Agent identities and their current status are roots of the trees, while their capabilities are presented as children of the trees. Optional tools for debugging purposes are provided by the GUI.
The platform is implemented in Java. An agent on the platform is a java class with private attributes that represent its beliefs, goals, basic actions and reasoning rules. This class has at least two methods, one for configuration of the 3APL object and one for running the object. The first method consists of parsing high-level specifications of agent beliefs, goals, actions, and reasoning rules, which are provided by the agent programmer, and representing them as private attributes. The second method is the implementation of the agent interpreter that determines the execution of the agent. The inter-platform communication was implemented with the use of the Java Agent Services API [3]. The Java Agent Services project is an initiative to define an industry standard for the development of network agent and service architectures.

References

[1] http://www.cs.uu.nl/3apl

[2] http://www.fipa.org

[3] http://www.java-agent.org

[4] http://sharon.cselt.it/projects/jade

[5] http://www.agent-software.com.au

[6] http://more.btexact.com/projects/agents/zeus/index.htm

ProAnita

A multi-agent solution for legitimate information retrieval

Femke de Jonge Nico Roos Pieter Spronck
Steven de Jong

IKAT, Universiteit Maastricht,
P.O. Box 616, NL-6200 MD, Maastricht

1 Introduction

The usage of electronic databases for the storage and retrieval of information is nowadays self-evident in many application areas. Nevertheless, when information is confidential and is distributed over several databases, gaining access to the required information is not free of problems. To illustrate some of these problems, consider the following example of a police detective investigating a murder case. Suppose that the detective has identified a suspect and wishes to find out if the suspect has been involved in other, related, crimes. Unfortunately, the detective does not know which database to consult (each police region has its own database and there are different databases for different topics), and he may not know how to handle the databases preferable to him. Moreover, police security officers will have to verify whether the detective is allowed to gain access to the requested information for this current crime investigation. This verification is based on legal norms, for example norms concerning the storage of information about persons.

To summarize, the usage of distributed databases containing valuable confidential information, requires a system that will improve the accessibility of information and the reliability and legitimacy of the information retrieval. The ANITA-project, part of the ToKeN2000 programme, aims at developing a multi-agent framework that enhances the application of norms for distributed access of information. ANITA (Administrative Normative Information Transaction Agents) is part of a collaboration between the Universities of Groningen[1], Utrecht[2], Leiden[3] and Maastricht[4]. The research of ANITA is generic, but will be applied on police and judicial intelligence domain.

ProAnita, the application described in this demonstration paper, is developed as a prototype for ANITA. It's purpose is to demonstrate the realizability of research goals of ANITA and to serve as a platform for experimenting during further ANITA research.

[1]Centre for Law & ICT: C.N.J. de Vey Mestdagh (Project Leader), P Dijkstra, Artificial Intelligence: L.R.B. Schomaker, W. Teepe

[2]ICS/Artifical Intelligence: J.-J. CH. Meyer, F. Dignum, H. Aldewereld

[3]eLaw@Leiden: H. Franken, L. Mommers

[4]IKAT: H.J. van den Herik, F. de Jonge, N. Roos, P. Spronck, S. de Jong, E. Smirnov

2 Design and functionality

ProAnita is a Multi-Agent System [5] consisting of four types of autonomous computer-applications (agents): the so-called IdentifierAgent, ManagerAgent, SearcherAgent and the AuthorizerAgent. The agents and their task in the agent-platform are depicted in figure 1.

The IdentifierAgent controls the platform on which the agents will be located. All agents have to announce themselves to the IdentifierAgent as they enter the platform. The IdentifierAgent checks the identity of an announced agent (through username and password), and then gives a certificate to the agent. The agent will use the certificate as a passport to identify himself while communicating to the other agents.

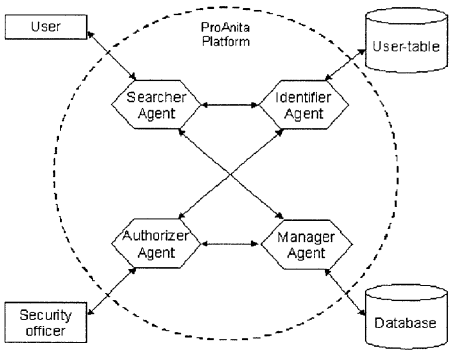


Figure 1: Schematic view of ProAnita

The SearcherAgent is operated by a user, for example a police detective, to perform queries or information requests. The SearcherAgent will send these queries or document requests to the corresponding information-managers (the ManagerAgents). The ManagerAgents check whether the requesting SearcherAgent or its user is allowed to view query-results. If so, the requested information or document is sent to the SearcherAgent which shows this to the user. If the SearcherAgent has not received permission to view the information, the reason of denial will be sent to the SearcherAgent as an answer to the request.

Each ManagerAgent controls a database on which queries can be performed. Moreover, a ManagerAgent has knowledge of (legal) norms, based on which the agent decides whether the information requesting user should receive access to the information. These norms, which are formulated as rules, can easily be adjusted in case of changes in the legal regulations.

The cases, in which the rules are not sufficient to pass a judgement concerning access to the information, the ManagerAgent will ask the AuthorizerAgent for authorization. The AuthorizerAgent is operated by a user who makes the concerning decision based on the users knowledge and the information received about the requesting party and its specific information request. The decision will be passed back to the SearcherAgent, which shows the decision and its motivation to the user.

[5] ProAnita is build on the JAVA platform Jade: http://sharon.cselt.it/projects/jade/ and runs on computers with Java Runtime Environment 1.4.1

Promedas: A diagnostic decision support system

Bert Kappen [a] Wim Wiegerinck [a] Ender Akay [a]
Marcel Nijman [a] Jan Neijt [b] André van Beek[b]

[a] SNN, Nijmegen University
[b] Department of Internal Medicine, University Medical Center Utrecht

Abstract

PROMEDAS is a medical patient-specific diagnostic decision support systems based on Bayesian networks. PROMEDAS aims to improve the quality and efficiency of the diagnostic process, while reducing its costs at the same time.

Diagnosis in internal medicine is a very complex process, requiring accurate patient data, a profound understanding of the medical literature and many years of clinical experience. Diagnosis is a process, by which a doctor searches for the cause (disease) that best explains the symptoms of a patient. The search process is sequential, in the sense that patient symptoms suggest some initial tests to be performed. Based on the outcome of these tests, a tentative hypothesis is formulated about the possible cause(s). Based on this hypothesis, subsequent tests are ordered to confirm or reject this hypothesis. The process may proceed in several iterations until the patient is finally diagnosed with sufficient certainty and the cause of the symptoms is established.

A significant part of the diagnostic process is standardized in the form of protocols. In the majority of the cases, these protocols are sufficiently accurate to make the correct diagnosis. For these 'routine' cases, a decision support system is not needed. In 10-20 % of the cases, however, the diagnostic process is more difficult. In these cases, normally a colleague is consulted that is particularly specialized in this disease or the literature is consulted. For these difficult cases computer based diagnostic decision support might be of added value.

PROMEDAS (PRObabilistic MEdical Diagnostic Advisory System) is a diagnostic decision support system that is intended to support diagnosis making in the setting of the outpatient clinic and for educational purposes. Its target-users are general internists, super specialists (i.e. endocrinologists, rheumatologists), interns and residents, medical students and others working in the hospital environment. Based on patient findings, it generates diagnostic advice in the form of a list of likely diagnoses suggestions for additional laboratory tests that are expected to be particularly informative to establish or rule out any of the diagnoses considered.

PROMEDAS is based on medical expert knowledge, acquired from the medical literature and textbooks by the medical specialists in our project team. The expert knowledge is stored in a data base, in such a way that extension and maintenance of the expert knowledge is easily facilitated. The data base specifies the causal relations between the variables

in the domain as well as statistics about these relations, such as sensitivities, specificities and prevalences. The sensitivities and specificities are the conditional probability for a test to be positive (negative) given that the diagnosis is true (false), respectively. From this database, the Bayesian network [1] and an interface for the PROMEDAS application are automatically compiled (see fig.).

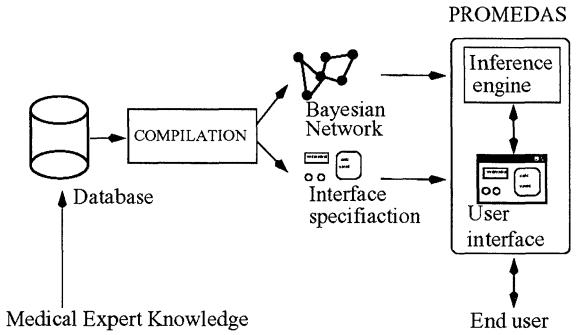


Figure 1: Organization of PROMEDAS development. End user may be connected via an Electronic Patient Dossier in future.

In the application, we use Bayesian inference to compute the probability of all diagnoses in the model given the patient data. Such computation can become prohibitive in particular due to the large number of parents that nodes can have. We use Noisy-OR relations [1] with the implementation as proposed by [2] to reduce the size of these tables. To further enhance efficiency, we first remove all unclamped leafs of the graph (i.e. findings that are not filled in) and do the computation based on this amputated model. As a result, the complexity of the inference task depends on the number of entered patient findings.

Currently, we have modeled a domain on lipid and vascular diseases. This domain consists of 70 diseases and approximately 400 findings.

Acknowledgements

This research was supported by the Technology Foundation STW.

References

[1] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, California, 1988.

[2] M. Takinawa and B. D'Ambrosio. Multiplicative factorization of noisy-MAX. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence UAI99*, pages 622–30, 1999.

Platform for Intelligent Agents on Embedded Devices: Project AgentLight

Fernando Koch[a]

[a] University of Utrecht, Intelligent Systems Group, Utrecht

Abstract

Project AgentLight [1] is an effort to take current Multi Agent Systems (MAS) theories into the expanding environment of handheld and embedded computational devices. Our goal is to create an integrated platform target to build multi-agent systems in embedded with limited memory, restricted computer power and narrow communication bandwidth.

1. Introduction

Project AgentLight [6] is an effort to take current Multi Agent Systems (MAS) theories into the expanding environment of handheld and embedded computational devices. With the arrival of Java 2 Micro Edition (J2ME) and its support by the industry now we can create our own applications that run on common placed pervasive computer devices, like mobile phones or PDAs.

In our architecture we use a specially designed First Order Logic machine having data and program logic represented in a homogeneous way. This is the main feature that allows us to exchange knowledge between different types of devices.

Some of the concepts behind this project are: infrastructure that works transparently and interconnected either in a desktop computer or a mobile phone; run on J2SE and J2ME; the minimum running environment would have 256Kb RAM memory; FIPA standards compliant.

2. System architecture

One other major requirement is to have a set of agents running in a single executable application. The multi-agent feature is useful when creating applications that require multiple distinct functionalities, which could be modularized one function per agent. We implemented the concept of 'Containers of agents', where the 'container' is basically and

aggregation of functionalities (Figure 2): 'Agent Manager'loaded with a 'basic agent management set of rules'; interface for communication infrastructure; the event scheduler; the Agents created dynamically during system lifetime.

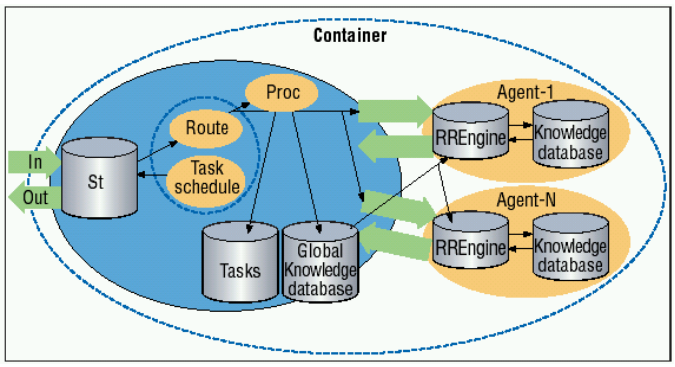
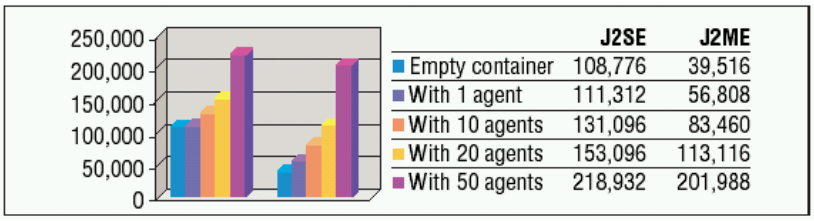


Fig. 2. System Architecture

The Container architecture allows us to create one single application with multiple agents in it. In our tests, we could create a 'knowledgeable' Container with 100 rules loaded using around 82Kb. The final conclusion is that it would be possible to create a multi-agent system with 10 'knowledgeable' agents using 166Kb of memory and a single-agent system with identical features using 82Kb.



	J2SE	J2ME
Empty container	108,776	39,516
With 1 agent	111,312	56,808
With 10 agents	131,096	83,460
With 20 agents	153,096	113,116
With 50 agents	218,932	201,988

Fig. 3. Memory usage for multi-agents results

References

[1] F. L KOCH, J-J. C. MEYER. Knowledge Based Autonomous Agents for Pervasive Computing using AgentLight. In: ACM/IFIP/USENIX International Middleware Conference, MIDDLEWARE 2003 WORK-IN-PROGRESS, IEEE Distributed Systems Online (http://dsonline.computer.org/0306/f/koc.htm). Brazil (2003).

Distributed planning of container terminal resources with agent technology.

Mark Leenaarts

Illyan BV, Sarphatistraat 642, 1018 AV Amsterdam

Abstract

The port of Rotterdam has initiated a research project (APPROACH) on distributed planning of barge vessel visits. Current operation procedures are not as good as they should. Unreliable planning's and long waiting times are getting a major concern.
The focus of this project will be on the challenge of improving the planning and scheduling of container terminal and barge operator resources. This will be done trough connecting and streamlining inter-enterprise processes and facilitating co-operation of involved parties taking into account their desire to keep business processes in their own hands. The presented solution entails assisted automated communication and planning implemented by the application of a multi-agent system.

1. Problem introduction

The time a barge will spend in Rotterdam is much longer than the time spend with the actual load and unloading movements. Although there is a initial slack of 60% in the sailing schedule, only 62% of the barges will eventually depart Rotterdam in time. The average length of a tour is 22 hours. The time spend one the actual loading and unloading movements is 7,5 hours[1].

In most cases a container barge will visit a number of terminals during a tour trough the port of Rotterdam. The barge-operator will create in conjunction with the barge skipper a schedule for the sequence of terminal visits. These schedules describe which terminals are visited, in what sequence, what the estimated times of arrival and departure at these terminals are and for each terminal how many containers have to be loaded and unloaded (moves). Constructing a schedule requires a tedious and time-consuming negotiation process.

The Terminal Operators (TO) and Barge Operators (BO) in Rotterdam are not willing to give full access to their own information sources. Competition issues and the supplier relation between BO's and TO's make this a reasonable proposition. This means that parties will demand the possibility to hide information for other participants. Also the domain is distributed and the

solution needs to follow the structure of the current situation. The domain dictates solution structure, not the other way around. This is, together with seamless integration in the existing system infrastructure, very important for acceptance in the field. To improve the efficiency of the planning process and the terminal and barge schedules, another requirement will be the support for a fast multi-channel communications infrastructure.

2. Proposed solution

Requirements:

- Automated multi-channel communications.
- Information hiding
- Support for planning and scheduling
- Facilitating automated negotiations
- Seamless integration
- Follows domain structure: distributed solution
- Personalisable strategies

A natural way of modelling and implementing the complex distributed domain of container handling is the application of multi-agent technology. Using a multi-agent architecture makes it possible to screen off enterprise information resources like order,- and schedule databases by a software agent. Each party will be modelled as an agent, this agent will implement enterprise specific strategies and will handle the message streams much faster then human operators will be able to. On the implementation level open, non-proprietary standards such as XML for communication and J2EE for enterprise connectivity will be used.

The demonstration will follow a representative showcase step by step and presents in 30 minutes an example of the value of multi-agent technology in a heterogeneous and distributed real world domain.

References

[1] Stichting RIL, CBRB, TNO-Inro; B-RIL Werkdocument, Afhandeling Containerbinnenvaart Rotterdam, 1998.

[2] PCR-RIL, Illyan APPROACH, Een onderzoek naar een gedistribueerde planning van barge afhandeling in de haven van Rotterdam, Rapportage fase 1, Rotterdam 2002.

Demonstration of Web Services Configuration

D. Richards[a] S. van Splunter[b] F.M.T. Brazier[b] M. Sabou[b]

[a] Macquarie University, Sydney 2109, Australia
[b] Vrije Universiteit Amsterdam, 1081HV Amsterdam

1. Introduction

Web service composition can provide a value-chain between customers and suppliers. The increasing number of services, and thus possible combinations, demands the development of dynamic and automatic techniques for their composition. Current commercial solutions are limited and are primarily static and manual. Automation requires reasoning about (semantic descriptions of) the services. In this demonstration we present our initial work [2] involving the semantic description of Web services using DAML-S [3] and how our Agent Factory [1] has used these descriptions in its design process to derive a Web service configuration.

2. Web Service configuration

Web Services are defined by the Stencil group as "loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols". The definition captures the self-contained, modular, composable and distributed nature of WSs. *DAML-S* is a DAML+OIL ontology conceptually divided into three sub-ontologies for specifying what a service *does,* how the service *works,* how the service is *implemented.* The *Agent Factory* has been developed as a servicing facility for automated re-design of software agents. The design process within the Agent Factory is one of configuration. This demonstration is configuring web-services described in DAML-S by an Agent Factory.

3. Demonstration

In [2] WS configuration is performed by the Agent Factory. Instead of agent components the Agent Factory configures WSs. The Agent Factory bases configuration on a template or design pattern, explicitly reasoning about requirements, the design process and the design artefact. The full paper [2] specifies a design trace of a configuration process in which existing web-services are combined. The DAML-S descriptions are the basis for the configuration process of which the resulting configuration is depicted in Figure 1.

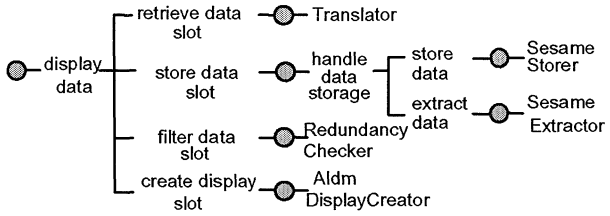


Figure 1. Resulting configuration of conceptual building blocks of WSs.

This demonstration features the prototype of this configuration process, supporting the work presented in [2].

Acknowledgements

This research is supported by NLnet (http://www.nlnet.nl). The authors also wish to thank Niek J.E. Wijngaards, and Oscar Scholten for their input.

References

[1] F.M.T. Brazier, and N.J.E. Wijngaards. Automated Servicing of Agents. *AISB Journal, Special Issue on Agent Technology*, 1:1, 5-20, 2001.

[2] D. Richards, S. van Splunter, F.M.T. Brazier, and M. Sabou. Composing Web Services using an Agent Factory. In *Proceedings of the AAMAS03 Workshop on Web Services and Agent-based Engineering*, 2003

[3] M. Sabou, D. Richards, and S. van Splunter. An experience report on using DAML-S. In *Proceedings of the WWW03 Workshop on E-Services and the Semantic Web*, Budapest, Hungary, May 2003.

Optimizing single-copy newspaper sales with Just Enough Delivery

Jan-Joost Spanjers Marco Bloemendaal Tom Heskes

SMART Research BV, P.O. Box 31070, 6503 CB Nijmegen
jed@smart-research.nl

Abstract

We present a software system called JED for optimizing newspaper sales. This software outperforms alternative systems, has a user friendly interface and is already used by various distributors.

1 Newspaper sales prediction

JED, for "Just Enough Delivery" is a software system for optimizing single-copy newspaper sales, based on a combination of neural and Bayesian technology. JED learns from the past, to predict the sales in the future. Bayesian technology is used for better learning (implementing a cross inference between the many different time-series involved), as well as for the cost-effective determination of deliveries based on sales predictions, accuracies, and sales strategies.

2 Performance

In comparative studies for Bauer-Verlag (Germany) and Edipresse (Switzerland), JED outperformed alternative approaches (see Figure 1).

Figure 1: JED's performance in three case studies. The total amount of actual deliveries, issued by the company's current system, and observed sales, based on these deliveries, are both indexed to 100 (crosses). Solid lines give the sales that could have been obtained if the model's suggestions had been followed for different amounts of deliveries. Dashed lines indicate the sales (↓) and delivery break-even points (←).

3 User friendly interface

Besides better prediction performance, JED contains many additional features: tools to select and target all outlets in specific regions and/or branches, algorithms to calculate and visualize different scenarios, graphics displaying the characteristics and explaining the predictions for individual outlets (see Figure 2), and so on. The new version is fully automatic, that is, predictions are continuously updated based on the latest available information.

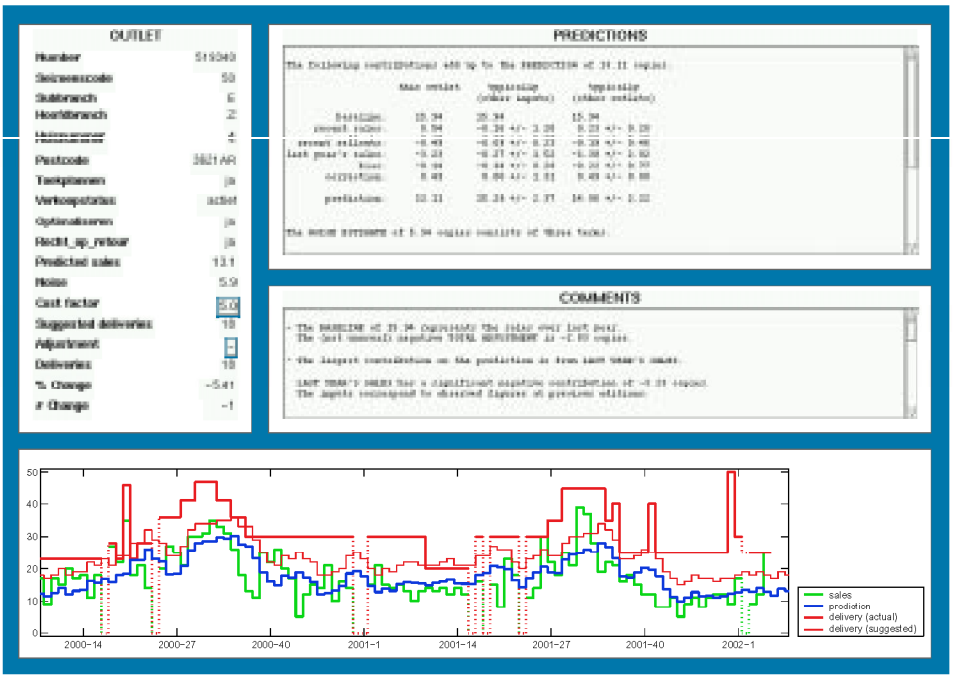


Figure 2: JED gui visualizing the properties of a particular outlet. It explains the predicted sales in terms of the explanatory variables and how this is translated into a suggested delivery. Based on this information, the operator can judge whether to accept or change the suggested delivery.

4 Implementations

JED 2.0 has been implemented at De Telegraaf and at Midesa (Portugal) for optimizing the distribution of the newspaper Público.

Index